Trabajo de Fin de Grado

Grado en Ingeniería Informática

Desarrollo de una aplicación de mensajería instantánea accesible para Android

Autor: Sergio Chicharro Sobrino

Tutoras: Ana María Iglesias Maqueda

Rocío Calvo Martín

Agradecimientos

A mis padres, por darme una educación y la oportunidad de formarme en el ámbito que me gusta.

A mi hermano, por ser un ejemplo a seguir y ayudarme en todo momento.

A Ana y Rocío, por su ayuda y coordinación en el proyecto.

A familiares y amigos, por su apoyo y su ánimo durante la realización del proyecto.

Resumen

Actualmente nos encontramos en una época en la que estamos la mayor parte del tiempo conectados con gente de nuestro entorno, y es muy sencillo comunicarnos con quien deseemos en cualquier momento y lugar, pero esta ventaja u oportunidad no está disponible para todo el mundo, o al menos no de forma óptima.

Disponemos de una gran variedad de aplicaciones en dispositivos móviles que nos son de gran utilidad para comunicarnos en cualquier momento, pero estas aplicaciones aún no están adaptadas completamente al uso y necesidad de todos los usuarios, pues hay usuarios con determinadas discapacidades que aún tienen verdaderos problemas para usar con facilidad y comodidad estas aplicaciones y beneficiarse de sus funcionalidades.

Es por esta razón por la que este Trabajo de Fin de Grado está orientado a acercar a estas personas aplicaciones que todo el mundo es capaz de utilizar con normalidad. La creación de una aplicación de mensajería instantánea accesible para dispositivos móviles que facilite el uso de estas herramientas a todos los usuarios, independientemente de sus capacidades.

Además, con este proyecto se busca la orientación de este tipo de aplicaciones a un entorno académico, agilizando las comunicaciones entre individuos de esta comunidad y haciendo llegar la información a su destino sea cual sea este y en cualquier momento.

Abstract

Nowadays we are in a time when we are most of the time connected with our people, and it's very easy to communicate with anyone where and when we want, but this advantage or opportunity is not available to everyone, or at least not in the best way.

We have a wide variety of applications for mobile devices that are very useful to communicate any time, but these applications are not fully adapted to the use and needs of every user, as there are users with certain disabilities who still have big problems to use easily and comfortable these applications and benefit from their functionalities.

It's because of this reason that this End of GradeProject is oriented to approach to these people applications that everyone is able to use normally. Creating an accessible instant messaging application for mobile devices that ease the use of these tools to every user, regardless of their abilities.

Furthermore, this project looks for the orientation of these kind of applications to an academic environment, making communications quicker between people of this community and bringing the information to its receiver wherever it is and whenever it's sent.

Contenido

Capítulo 1
Introducción

1.1 Objetivos

El objetivo de este trabajo de fin de grado es el análisis de las tecnologías y el posterior uso de las mismas para el desarrollo de una aplicación de mensajería instantánea accesible compatible con dispositivos móviles con tecnología Android.

La causa del proyecto es acercar la tecnología móvil a usuarios con deficiencias que dificulten la utilización de estas herramientas, hacer que estas herramientas resulten cómodas y rápidas para todo tipo de usuarios, facilitando, en este caso, la comunicación entre una comunidad, independientemente de sus condiciones físicas y psicológicas.

Además, se pretende dar una orientación educativa al proyecto, es decir, convertirlo en una herramienta académica accesible que mejore y facilite la comunicación y el intercambio de información entre los miembros de la comunidad educativa.

El desarrollo de la aplicación se puede dividir en 3 objetivos más concretos y técnicos:

1. **Desarrollo de la aplicación móvil.**
   Desarrollo de la parte de la aplicación en un dispositivo móvil. Despliegue de una interfaz gráfica intuitiva para todos los usuarios, capaz de almacenar y conectarse con una base de datos, capaz de lanzar avisos al usuario cuando haya algún tipo de novedad en la aplicación y capaz de conectarse e interactuar con el servidor web.

2. **Desarrollo del servidor web.**
   Desarrollo de un entorno web capaz de recibir peticiones de la aplicación cliente y en función de estas, enviar una respuesta. Este servidor web, tiene que estar alojado en un dominio operativo en todo momento para que se pueda acceder a él desde la aplicación.
   Desde el servidor se podrá acceder a una base de datos global de la aplicación, consultando, insertando, modificando o suprimiendo datos cuando sea necesario.

3. **Despliegue de la base de datos global.**
   Para el correcto funcionamiento de la aplicación deberemos disponer de una base de datos alojada en un servidor que este continuamente disponible, para poder usarla en cualquier momento de la ejecución de nuestra aplicación.

4. **Configuración y sincronización de servicios.**
   Para la utilización de la aplicación será necesaria la disposición de un servicio así como su configuración con nuestra aplicación cliente, para que el acceso al servidor sea posible.
   Además, utilizaremos un servicio de Google (GCM) para mejorar el rendimiento de la aplicación. Por esta razón, será necesario configurar este servicio con todo nuestro proyecto, para darle acceso al mismo.

1.2 Estructura del documento

En este apartado detallaremos como está estructurado el documento:

- **Capítulo 1 : Introducción**
  Breve introducción del proyecto que ayudará al lector a situarse en el contexto y le dará una visión general del proyecto, facilitando su posterior desarrollo.

- **Capítulo 2 : Gestión del proyecto**
  En este capítulo se explicará el proceso de planificación y evolución del proyecto. Define la organización y control de los recursos utilizados para el desarrollo del proyecto y todas sus fases.

- **Capítulo 3 : Estado del arte**
  Conformado por un análisis de las tecnologías actuales que afectan a el desarrollo de nuestro proyecto. Desde la tecnología a utilizar, Android; hasta las necesidades de accesibilidad que deberemos implementar y la funcionalidad de carácter académico necesaria.

- **Capítulo 4 : Análisis de las tecnologías**

En este apartado se analizarán y explicarán detalladamente todas las tecnologías que se han tenido en cuenta para el desarrollo de la aplicación, las decisiones tomadas en cuanto al uso de estas y las razones por las cuales estas decisiones fueron tomadas.

- **Capítulo 5 : Desarrollo de la aplicación**
  Describe el desarrollo y la programación de toda la aplicación explicando cada función de la misma, permitiendo, por un lado, saber cómo utilizar la aplicación y, por otro lado, conocer cómo funciona y como está desarrollada la aplicación .

- **Capítulo 6 : Análisis de costes**
  Descripción y análisis de los costes asociados al desarrollo del proyecto.

- **Capítulo 7 : Conclusiones y trabajos futuros**
  Desarrollo de las conclusiones obtenidas tras la realización del proyecto y sobre este y exposición de los proyectos futuros en relación al proyecto.

- **Capítulo 8 : Glosario de términos y abreviaturas**
  En este apartado se explicarán todos los términos referentes al proyecto que pertenezcan a una jerga especifica del proyecto, tecnicismos, nombres de herramientas o servicios y abreviaturas para el correcto entendimiento del documento.

- **Capítulo 9 : Bibliografía**
  Lista de las diferentes fuentes de información utilizadas para la elaboración del proyecto con información relativa a estas, así como el título del documento, el enlace...

- **Capítulo 10 : Anexos**
  Enumeración de los distintos documentos que serán adjuntados al presente documento como complemento de este y para facilitar el entendimiento del proyecto y la aplicación.

Capítulo 2
Gestión del proyecto

2.1 Gestión software: Metodología y ciclo de vida

Establecer una metodología o un estilo de trabajo para nuestro proyecto es importante pues nos ayudará a estructurar nuestra forma de trabajar y planificar las siguientes tareas a realizar dentro del proyecto.

Al tratarse de un proyecto de desarrollo, será necesario el conocimiento de todas las herramientas y lenguajes que se vayan a utilizar para la elaboración del proyecto. Además de esto, será necesario el conocimiento de los requerimientos y conceptos relativos a toda aplicación desarrollada en Android, el estado actual de la accesibilidad en herramientas como la que vamos a desarrollar y el estado de la comunicación dentro de la comunidad educativa.

Es importante un gran conocimiento de estos puntos, pues para la realización del proyecto será necesario adaptar la herramienta a unas guías, normas y aspectos legales actualmente vigentes.

Teniendo en cuenta toda la información acerca de estos aspectos, determiné un ciclo de vida siguiendo un modelo de cascada (Figura 1); uno de los modelos de ciclo de vida más utilizados para la realización de proyectos software.

Este modelo ha sido utilizado en cada una de las fases en las que está dividido el proyecto. Este enfoque del proyecto establece la obligación de finalizar una etapa del ciclo de vida antes de comenzar la siguiente y nos permite regresar a una etapa anterior en el caso de detectar algún fallo en la actual, pero esto conllevaría la repetición de las etapas posteriores a la que regresamos.

Esta metodología se adapta a la forma de trabajo dentro de un proyecto de desarrollo de software, pues un cambio en una de las etapas del ciclo conllevará, la gran mayoría de las veces, cambios y modificaciones en etapas anteriores de la fase.

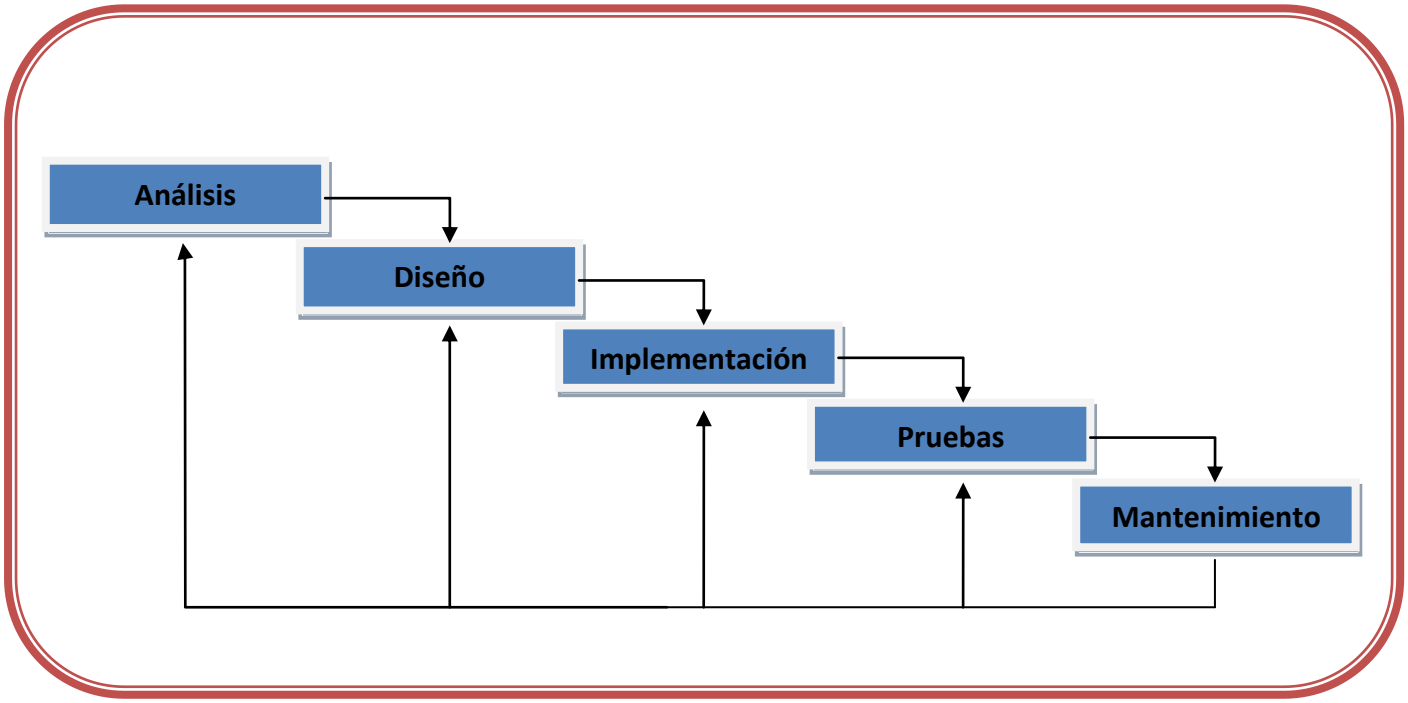

Figura 1: Metodología en cascada

2.2 Organización del trabajo

2.2.1 Esquema WBS (paquetes de trabajo)

Para la organización del desarrollo del proyecto, este se ha dividido de acuerdo a una descomposición jerárquica en paquetes de trabajo WBS (Work Breakdown Structure). De esta manera el proyecto resultará más fácil de abordar y tendremos en todo momento de su desarrollo constancia del avance alcanzado y el trabajo que resta realizar (Figura 2).

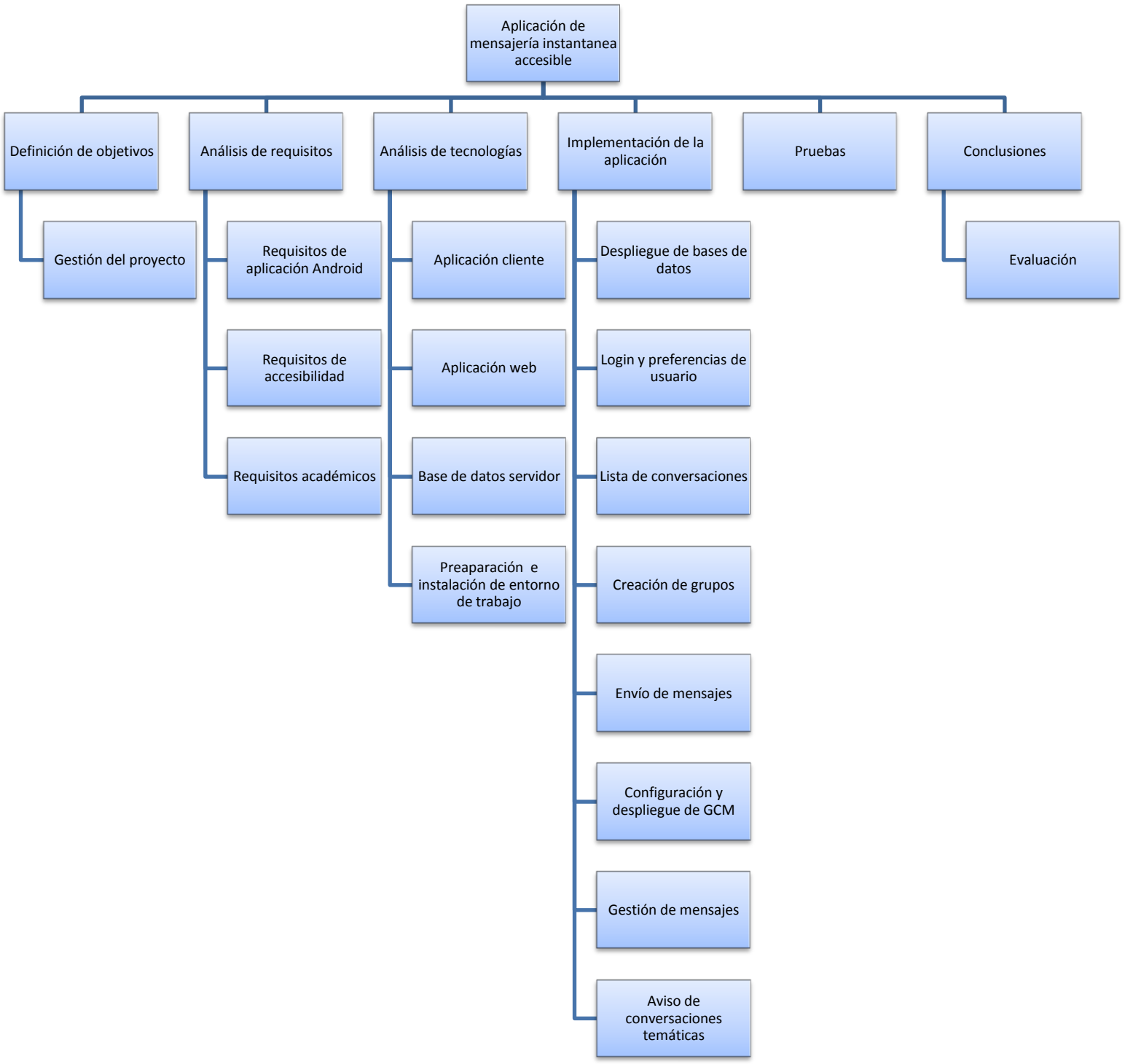

Figura 2: Estructura WBS del proyecto

2.3 Planificación inicial

2.3.1 Secuencialidad de los trabajos (PERT)

La planificación del proyecto se llevará a cabo y se ilustrará utilizando un diagrama PERT (Figura 3). En este diagrama podremos distinguir las grandes fases del proyecto así como su orden de desarrollo y elaboración.

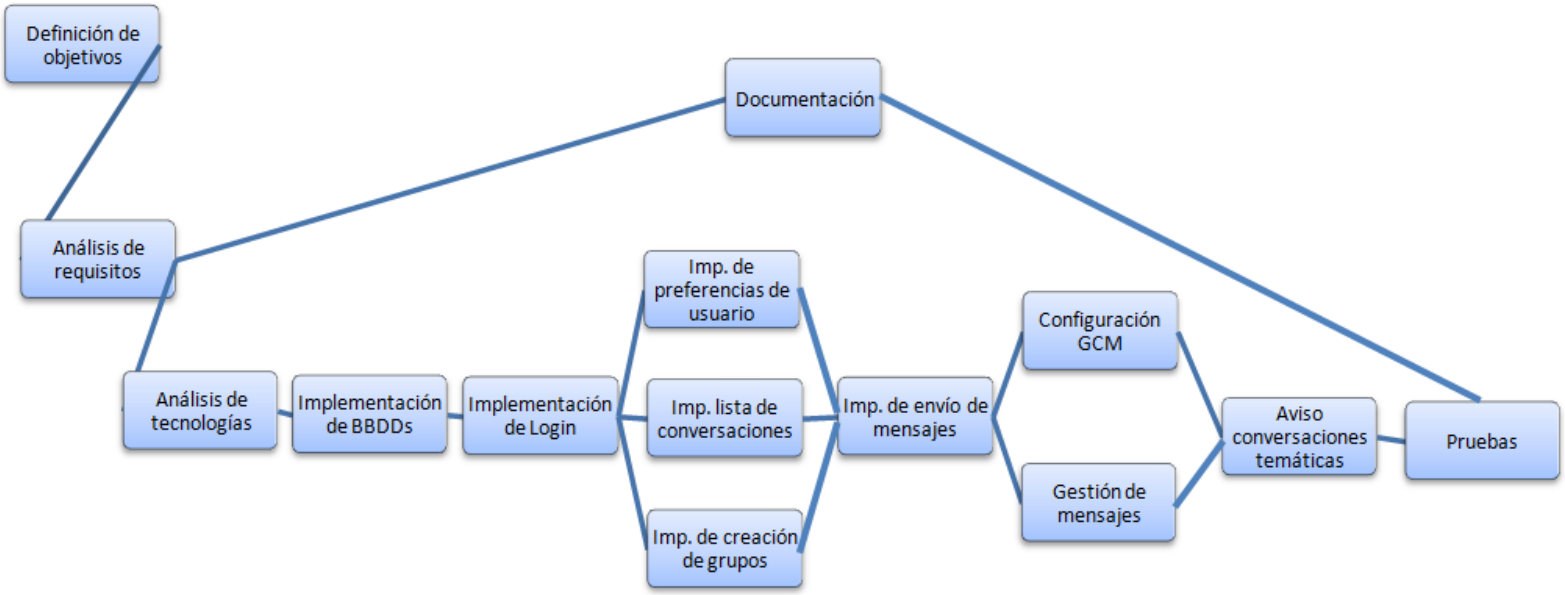

Figura 3: Diagrama PERT

2.3.2 Planificación temporal (GANTT)

Para facilitar el desarrollo y entendimiento global del proyecto se desarrollo una planificación en base al tiempo mediante la utilización de un diagrama de Gantt, en el cual podremos ver todas las tareas del proyecto y su duración(Figura 4).

Lamentablemente y debido a las dimensiones de dicho diagrama, mostrarlo en el presente documento supone gran dificultad, por esta razón, el diagrama además será entregado como anexo con este documento.

	Task Name	Start Date	End Date	Duration
1	**Aplicación de mensajería instantánea accesible para Android**	**15/03/14**	**19/06/14**	**97**
2	Definición de objetivos	15/03/14	17/03/14	3
3	Análisis de requisitos y documentación	18/03/14	25/03/14	8
4	Análisis de aplicaciones Android	18/03/14	20/03/14	3
5	Análisis de accesibilidad en aplicaciones Android	21/03/14	24/03/14	4
6	Requisitos académicos	25/03/14	25/03/14	1
7	Análisis de tecnologías a utilizar en el desarrollo	26/03/14	03/04/14	9
8	Herramientas para el desarrollo de la aplicación cliente	26/03/14	27/03/14	2
9	Herramientas para el desarrollo de la aplicación web	28/03/14	29/03/14	2
10	Herramientas de desarrollo de la base de datos del servidor	30/03/14	31/03/14	2
11	Instalación de las herramientas seleccionadas	01/04/14	03/04/14	3
12	Implementación de la aplicación	04/04/14	06/06/14	64
13	Bases de datos	04/04/14	10/04/14	7
14	Diseño de base de datos cliente	04/04/14	05/04/14	2
15	Diseño de base de datos servidor	06/04/14	07/04/14	2
16	Implementación y despliegue de bases de datos	08/04/14	10/04/14	3
17	Login	11/04/14	19/04/14	9
18	Desarrollo de la pantalla de login	11/04/14	11/04/14	1
19	Desarrollo de la pantalla de creación de cuenta de usuario	12/04/14	12/04/14	1
20	Desarrollo de la pantalla de recuperación de cuenta	13/04/14	13/04/14	1
21	Implementación de la lógica en las interfaces	14/04/14	17/04/14	4
22	Implementación de peticiones en la parte del servidor	18/04/14	19/04/14	2
23	Preferencias de usuario	20/04/14	24/04/14	5
24	Interfaz gráfica de las preferencias de usuario	20/04/14	22/04/14	3
25	Lógica y almacenamiento de preferencias	23/04/14	24/04/14	2
26	Lista de conversaciones	20/04/14	24/04/14	5
27	Interfaz de creación de listas dinámicas	20/04/14	21/04/14	2
28	Acceso a base de datos local y funcionalidad de la interfaz	22/04/14	24/04/14	3
29	Creación de grupos	20/04/14	09/05/14	20
30	Interfaz y funcionalidad de pantalla para crear grupos	20/04/14	24/04/14	5
31	Interfaz y funcionalidad de información de un grupo	25/04/14	30/04/14	6
32	Interfaz y funcionalidad de consulta y agregación de participantes de un grupo	01/05/14	04/05/14	4
33	Interfaz y funcionalidad de agregación de contactos	05/05/14	06/05/14	2
34	Implementación de comunicación y peticiones en el servidor	07/05/14	09/05/14	3
35	Envío de mensajes	20/05/14	27/05/14	8
36	Interfaz y funcionalidad de la pantalla de conversación	20/05/14	21/05/14	2
37	Interfaz y funcionalidad de la pantalla de información de contacto	22/05/14	23/05/14	2
38	Interfaz y funcionalidad de abrir conversación	24/05/14	25/05/14	2
39	Implementación de peticiones en el servidor	26/05/14	27/05/14	2
40	Configuración y despliegue de GCM	28/05/14	31/05/14	4
41	Configuración del servicio en Google Developers	28/05/14	28/05/14	1
42	Implementación del registro del dispositivo al servicio	29/05/14	29/05/14	1
43	Implementación de el envío de datos del servidor al servicio GCM	30/05/14	30/05/14	1
44	Implementación de la clase en la aplicación receptora de GCM	31/05/14	31/05/14	1
45	Gestión de mensajes	28/05/14	31/05/14	4
46	Implementación del buscador y filtro de mensajes	28/05/14	29/05/14	2
47	Implementación del marcador de mensajes	30/05/14	31/05/14	2
48	Aviso de conversaciones temáticas	05/06/14	06/06/14	2
49	Implementación de la clase servicio	05/06/14	06/06/14	2
50	Pruebas y detección y corrección de errores	17/06/14	19/06/14	3
51	Documentación	07/06/14	16/06/14	10
52	Elaboración de la memoria	07/06/14	12/06/14	6
53	Elaboración de memoria-resumen en inglés	13/06/14	14/06/14	2
54	Elaboración de manual de usuario	15/06/14	15/06/14	1
55	Elaboración de manual de desarrollador	16/06/14	16/06/14	1

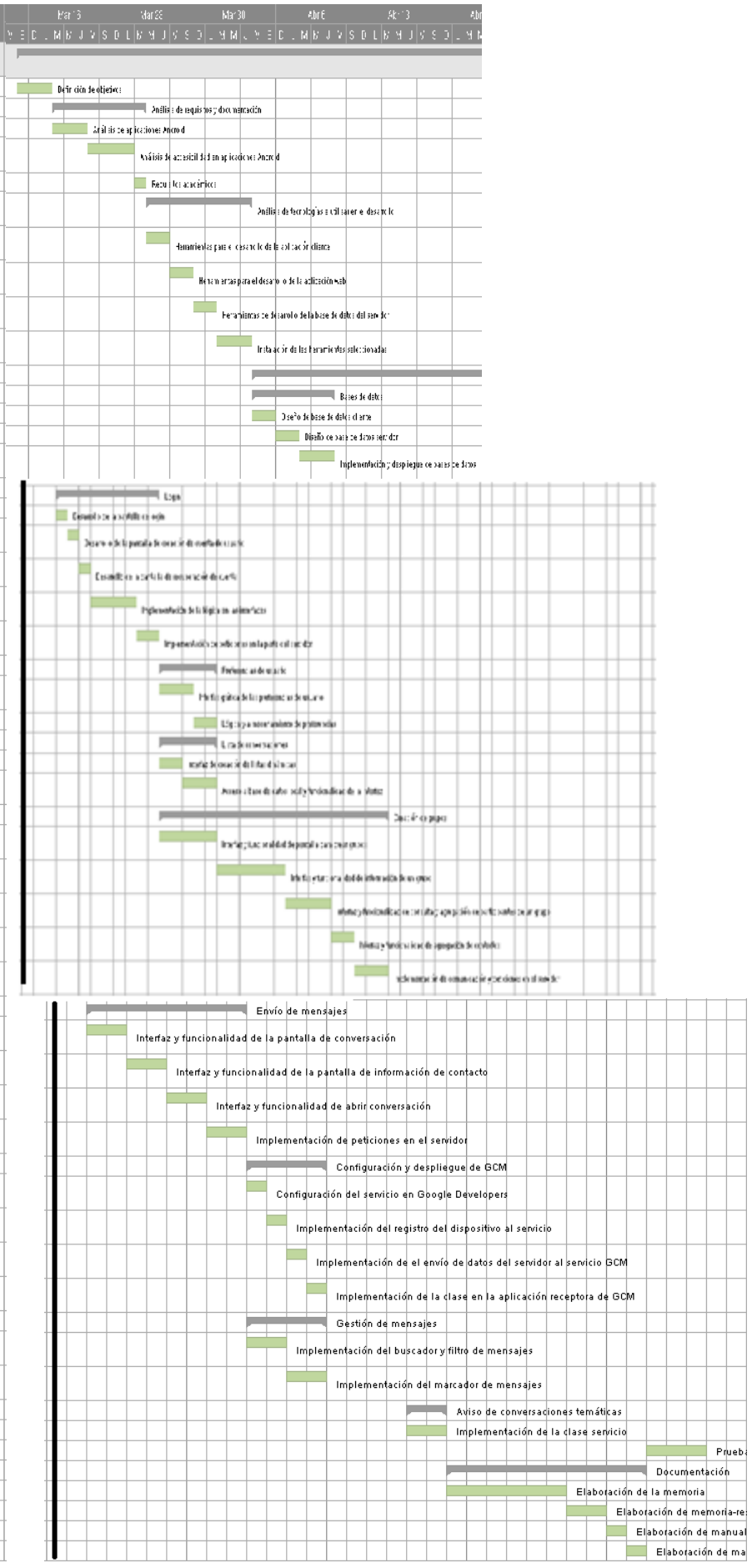

Figura 4: Diagrama GANTT

Capítulo 3
Estado del arte

3.1 Introducción

El objetivo principal del proyecto es el desarrollo de una aplicación de mensajería instantánea para dispositivos móviles, en concreto, para dispositivos móviles con sistema operativo Android. Además, esta aplicación será accesible, es decir, tendrá la funcionalidad y adaptabilidad necesarias para que todas las personas, y especialmente las que tengan alguna discapacidad física o psicológica, sean capaz de utilizar la misma y beneficiarse de sus funciones.

La aplicación también tendrá una orientación académica, con la que profesores de la universidad serán capaces de enviar comunicados a los alumnos de forma instantánea, crear conversaciones para tratar temas sobre asignaturas o concertar tutorías, entre otras funciones.

3.2 Requisitos básicos

Todas las aplicaciones deben contener unas características generales para facilitar su correcta utilización y facilitarla a cualquier tipo de usuario. Dado que no existe ninguna normativa específica que regule el desarrollo de aplicaciones accesibles, estos requisitos estarán basados en las recomendaciones que nos proporciona la guía de desarrolladores de Android para aplicaciones accesibles (http://developer.android.com/intl/es/guide/topics/ui/accessibility/checklist.html). Toda aplicación debería adaptarse a los puntos redactados a continuación:

- Navegación intuitiva. Desarrollar una interfaz y su funcionalidad de forma intuitiva, de forma que el usuario pueda prever la acción que se va a realizar en la aplicación.

- Usar un tamaño de los objetos de interacción recomendado. Seleccionar un tamaño para los objetos que llevarán a cabo una funcionalidad, que permita al usuario tocarlo sin problema.
- Etiquetar el contenido de la interfaz de usuario significativamente. Proporcionar información sobre los elementos de la IU que permita al usuario comprender el funcionamiento de los mismos o la información que proporcionan.
- Proporcionar alternativas para funcionalidad volátil. Desarrollar funcionalidad alternativa a funcionalidades que puedan no estar disponibles en algún momento de la ejecución de la aplicación.
- No informar de cambios en la aplicación solo con audio. Esta notificación de que se ha producido algún cambio en la ejecución de la aplicación debe estar acompañado de otro efecto informativo.
- Personalizar control de vistas. Si creamos un control de la interfaz personalizado, deberemos asegurarnos de que este es accesible y proporcionar descripciones de sus elementos.

3.2 Accesibilidad orientada a personas con discapacidad

Para llevar a cabo el desarrollo de la aplicación, primero es necesario analizar las necesidades del público al que está orientado dicha aplicación. En nuestro caso, la aplicación está orientada especialmente a personas con discapacidades en las cuales la tecnología actual nos permite desarrollar funciones de apoyo, aunque no excluye al resto de poder utilizarla; por tanto, debemos tener en cuenta que, además de cumplir con los requisitos y estándares que cualquier aplicación en Android debe incluir, deberemos tener en cuenta aquellos que permitan utilizar la aplicación a personas con discapacidades.

Estos estándares anteriormente nombrados han sido proporcionados por la coordinadora desde el siguiente enlace http://www.ati.es/IMG/pdf/CalvoVol8Num1.pdf. A continuación encontraremos una lista de los mismos junto con una breve descripción de los problemas de discapacidad a los que están orientados estos documentos:

- **WCAG_2.0_2.2.1_IR, WCAG_2.0_1.1.1_CR, MWBP_1.0_36_CR, UDL_2.0_1.3_CR.**
  Estos estándares están dirigidos a problemas de visión y a interactuar con el dispositivo en lugares con mucha luz.

- **WCAG_2.0_1.1.1_CR, MWBP_1.0_36_CR, UDL_2.0_1.3_CR, MWBP_1.0_25_CR.**
  Dirigidos a problemas de visión debidos al pequeño tamaño de los elementos de la pantalla.
- **WCAG_2.0_2.21_IR, WCAG_2.0_2.2.2_IR, MWBP_1.0_14_IR, IMS_v2_5_IR, IMS_V2_6_IR, IMS_v2_7_IR.**
  Dirigidos a problemas motrices y de movilidad.
- **WCAG_2.0_1.3.2_CR, WCAG_2.0_1.3.1_CR.**
  Normativas dirigidas a solventar problemas de aprendizaje.
- **WCAG_2.0_2.2.1_IR, WCAG_2.0_2.2.2_IR, MWBP_1.0_14_IR, IMS_v2_5_IR, IMS_v2_6_IR, IMS_v2_7_IR, WCAG_2.0_1.1.1_CR, MWBP_1.0_36_CR, UDL_2.0_1.3_CR, WCAG_2.0_1.3.1_CR, WCAG_2.0_1.3.2_CR.**
  Dirigidos a resolver funciones de control de flujo de la información por parte del usuario dentro de la aplicación.

Con este fin, analizaremos las diferentes discapacidades que pueden afectar en el uso de nuestra aplicación. Estos tipos de discapacidades han sido diferenciados bajo un criterio propio, en función de los productos de apoyo (funcionalidad extra que ayudará a las personas con discapacidad en el uso de la aplicación) a desarrollar sobre la funcionalidad base de la aplicación para adaptarse a estos distintos factores:

3.2.1 Discapacidad sensorial

- **Discapacidad visual.**

  La discapacidad visual se debe a la limitación total o parcial de la visión. Realmente, hay muchos tipos de deficiencia visual, pero poniéndonos en el peor de los casos(limitación total de la visibilidad) podemos solventar también el resto de discapacidades visuales más leves.
  Para ayudar a personas con discapacidad visual a utilizar nuestra aplicación, debemos proporcionar un producto de apoyo a la interfaz visual que nos permita informar al usuario del estado actual de la aplicación. Utilizaremos funcionalidades como:
  Talk-back: Servicio proporcionado por Google el cual nos permitirá incluir en nuestra aplicación un *feed back* mediante audio, que describa la funcionalidad y los procesos que se esté realizando dentro de la aplicación.

Explore by touch: Característica del sistema que nos informará del contenido de la interfaz cuando toquemos la pantalla en uno de los elementos de la interfaz.
Además la aplicación facilitará la personalización de las preferencias del usuario, así como tamaño, color o fuente de letra para ayudarle a utilizarla en el caso de que no tenga una limitación total de la visibilidad.

- **Discapacidad auditiva.**

La discapacidad auditiva se debe a la limitación total o parcial de la función auditiva. Al igual que la discapacidad visual, existen varios grados de deficiencia auditiva, sin embargo, previendo la correcta utilización de la aplicación en el peor de los casos, dicha funcionalidad extra subsanará discapacidades auditivas más leves.
En este caso, es mucho más sencillo, pues normalmente toda la funcionalidad puede llegar a comprenderse de forma visual. Sin embargo, las aplicaciones de mensajería instantánea, al recibir un mensaje o una notificación, suelen informar al usuario emitiendo sonido desde el dispositivo. Para evitar el problema de que personas con esta discapacidad no sean capaces de percibir tal notificación, se emitirá a su vez, desde el dispositivo, una luz led cuyo color podrá ser personalizable por el mismo usuario; y la posibilidad de hacer que el dispositivo vibre de forma que el usuario sea capaz de percibir la notificación de una forma alternativa o añadida a la luz.

3.2.2 Discapacidad física o motriz

La discapacidad física en brazos y manos por falta de miembros, limitación de fuerza o problemas de coordinación puede afectar al correcto uso de nuestra aplicación, es por ello que, con el fin de facilitar su utilización, serán empleados e implementados gestos simples y fáciles de realizar para llevar a cabo la ejecución de la aplicación.

Además podremos utilizar un sistema de dictado de mensajes, que facilitará y agilizará a individuos con estas discapacidades la escritura y envío de los mismos, haciéndolos exentos del uso de las manos para este fin.

3.2.3 Discapacidad cognitiva, intelectual y neurológica

Este tipo de discapacidades también afectan al correcto uso de la aplicación, dificultando la capacidad para utilizarla. Además, conllevan una gran dificultad a la hora de procesar cualquier tipo de información. Es por esta razón que, si queremos que la aplicación sea accesible para personas con este tipo de discapacidades, la adaptemos, librando la pantalla en todo momento de sobrecargas de información, mostrando una interfaz clara que provoque las menos confusiones de funcionalidad posibles; y deberemos proporcionar toda la información necesaria para el correcto entendimiento de cada elemento de la interfaz, siendo posible consultarlo en cualquier momento.

3.3 Mensajería instantánea para el aprendizaje

Existen numerosas aplicaciones de mensajería instantánea, pero pocas de estas cuentan con las prestaciones necesarias para que se las considere una aplicación accesible para todo tipo de usuarios. El propósito de este proyecto es, además de desarrollar una aplicación que sea accesible, adaptarla y dotarla de una funcionalidad para su uso académico.

Ya que ninguna de las aplicaciones actuales de mensajería instantánea más importantes del momento cuenta con una funcionalidad para un campo en concreto, hemos querido que este proyecto esté orientado a la educación, facilitando la comunicación entre personas que compartan algún interés educativo (una misma asignatura, un mismo proyecto...) ; dando acceso a la aplicación a los mismos profesores también, haciendo que la comunicación entre profesores y alumnos sea más rápida y más directa.

Para dotar a la aplicación de un enfoque educativo se desarrollará toda la funcionalidad expresamente para dicho fin. Así, desde la aplicación será posible realizar acciones que actualmente con la tecnología de la que disponemos conllevaría algo más de tiempo como:

- **Realización de consultas:** Sería posible realizar una consulta acerca de una práctica o una duda sobre algún tema que comprenda la asignatura en cuestión, a cualquier profesor en un momento.

- **Creación de grupos temáticos:** Los profesores serán capaces de crear grupos acerca de temas de la asignatura que deban ser tratados con mayor profundidad, para debatir sobre los mismos o aclarar dudas que puedan haber surgido.

- **Concertación de tutorías:** Gracias a la aplicación, se podrán concertar rápidamente tutorías con los alumnos para tratar temas académicos (revisiones de examen, dudas sobre la asignatura, defensa de prácticas...).

3.4 Normas, estándares y aspectos legales

Para el desarrollo de nuestra aplicación Android deberemos tener en cuenta diferentes aspectos relativos a las aplicaciones Android y la accesibilidad en estas. La aplicación final deberá cumplir con una serie de requisitos y normas que garanticen el correcto funcionamiento de la aplicación y que su accesibilidad cumpla con su cometido.

Con este mismo fin, el consorcio internacional World Wide Web Consortium produjo recomendaciones básicas que se deberían tener en cuenta a la hora de desarrollar una aplicación móvil.

A continuación, describiré estas recomendaciones que, aunque no tienen validez legal, si es conveniente cumplir para el correcto uso y aceptación de nuestra aplicación:

- **Problemas de presentación.** Al desarrollar una aplicación en un dispositivo móvil no enfrentamos al problema de que tenemos un reducido espacio donde presentar la información, además, este espacio puede ser variable, pues nuestra aplicación podrá utilizarse en numerosos dispositivos con diferente resolución de pantalla. Es necesario controlar que la interfaz de nuestra aplicación y cada uno de sus elementos sea visible en la pantalla de cada dispositivo en el que pueda utilizarse.
- **Entradas.** Al igual que el punto anterior, el tamaño de pantalla también influir en los elementos interactuables de la misma, es por esto que deberemos asegurarnos de que cada botón e input tiene tamaño suficiente para ser utilizado.
- **Ancho de banda y coste.** Un dispositivo móvil dispondrá de una red y una conexión bastante lenta en la mayor parte de los casos. Es por esta razón que deberemos limitar y optimizar el uso de la red en nuestra aplicación, sin sobrecargas de flujo de datos, ni descarga de archivos pesados, pues podrían provocar irregularidades en la aplicación y el dispositivo.

- **Uso de publicidad.** Es posible incluir publicidad y anuncios dentro de nuestras aplicaciones, pero esto también puede afectar negativamente al uso de nuestra aplicación si lo utilizamos de forma abusiva. En el caso de incluir publicidad en nuestra aplicación debemos procurar que no ocupen gran parte de la pantalla ni dificulten el funcionamiento de la aplicación aunque la publicidad es un elemento preferiblemente prescindible.

Capítulo 4
Análisis de las tecnologías

4.1 Aplicación cliente

El desarrollo de la aplicación de mensajería se dividirá en dos partes: una parte que residirá en el dispositivo móvil, la parte del cliente; y otra parte que estará alojada en la web, la parte del servidor. En esta sección explicaré todo lo relacionado con la tecnología utilizada para el desarrollo de la aplicación cliente y las decisiones tomadas acerca de la misma.

4.1.1 ¿Por qué Android?

Para desarrollar en proyecto he elegido el sistema operativo Android. Esta elección ha sido justificada por el hecho de que Android es una plataforma abierta y de código abierto, características que lo hacen mucho más atractivo a la hora de desarrollar la aplicación y te facilitarán la implementación de la misma.

Además, Android es el sistema operativo más utilizado en Europa actualmente en dispositivos móviles, y el que poseen la mayoría de los mismos. Es por esto que también te anima a desarrollar con él, pues el público que tendrá al alcance tu aplicación será mucho mayor. La afirmación de que el sistema operativo Android es el

más utilizado actualmente se basa en un estudio de mercado realizado por la compañía internacional Kantar Worldpanel (dedicada al estudio del mercado) que afirma que el sistema operativo Android fue el más vendido en Europa a finales de 2013 (http://www.kantarworldpanel.com/global/News/Android-ends-the-year-on-top-but-Apple-scores-in-key-markets)(Figura 5).

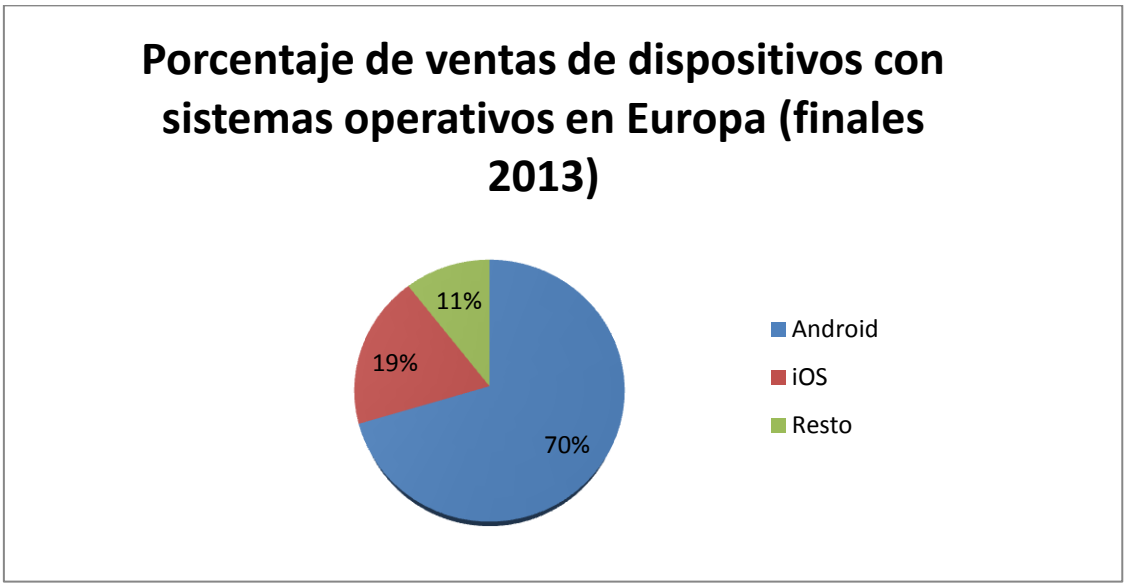

Figura 5: Gráfico de venta de sistema operativos para dispositivos móviles

Otra de las razones de la elección de Android es que es un sistema multitarea, lo que nos ofrecerá muchas más posibilidades a la hora de incluir funcionalidades a nuestro proyecto; posibilidades que un sistema que no sea multitarea no nos podría ofrecer.

Por último, otro de los puntos a favor de la elección de esta tecnología es el hecho de su gran comunidad de desarrolladores. Hay infinidad de documentación sobre la programación en Android en internet, lo que nos dará la oportunidad de resolver cualquier problema que podamos tener mientras desarrollemos o incluso incluir funcionalidad extra basada en toda esta documentación.

Para desarrollar en Android se pueden utilizar varios lenguajes de programación Java, C #, C, .NET, VisualBasic... Sin embargo, hemos utilizado Java para el desarrollo de la aplicación, pues es el lenguaje nativo del sistema operativo, además de ser el lenguaje utilizados para el desarrollo de las APIs de Google, lo que nos vendrá bien para la compatibilidad de estas con nuestra aplicación.

4.1.2 Entorno de desarrollo

Para el desarrollo de la aplicación cliente del proyecto se ha utilizado Eclipse, un sistema de herramientas de programación de código abierto para desarrolladores mayormente utilizado para programar en Android aunque también habilita soporte para otros lenguajes como C++, perl, PHP...

Para acondicionar dicho entorno para la programación de aplicaciones Android necesitaremos instalar el conjunto de herramientas SDK (Software Development Kit). Este nos ofrece la oportunidad de depurar nuestra aplicación, también podremos emularla en un simulador de dispositivo móvil, haciendo más cómodas las pruebas sobre la aplicación. Además el SDK Manager nos permitirá descargarnos paquetes y librerías que nos podrán ser de mucha utilidad al desarrollar nuestra aplicación, como por ejemplo las APIs de Google.

Además deberemos instalar en Eclipse el ADT (Android Development Tools), un plugin que se integra con el entorno de Eclipse y que nos ofrecerá muchas funciones para ayudarnos en el desarrollo de la aplicación.

Antes de elegir el IDE en el que iba a desarrollar el proyecto, evalué diferentes opciones como son Netbeans, Eclipse o Android Studio.

- **Netbeans** es una herramienta muy utilizada al igual que Eclipse, sin embargo, Netbeans no permite realizar el diseño de las interfaces de la aplicación, lo que para una aplicación en Android es realmente importante, pues supone una gran parte del desarrollo; por otro lado Eclipse si cuenta con estas herramientas (al instalar ADT) por lo que desarrollar con Eclipse es más recomendable en este caso.

- **Android Studio** es un IDE relativamente nuevo, orientado expresamente al desarrollo en Android, con el que seguramente, podríamos conseguir mejores resultados en nuestra aplicación y de forma más sencilla y rápida; además este IDE esta desarrollado por Google, con lo que podemos aventurar que será un producto de confianza. Por otro lado, este software aún está en una fase muy temprana de desarrollo (recientemente alcanzó la fase 0.60 de desarrollo) y ellos mismo informan de que este IDE aún es una *preview* de lo que será en un futuro.

Por estas razones he creído conveniente que la mejor opción era desarrollar la aplicación con Eclipse, ya que es un IDE con el que he trabajado mucho tiempo y Android Studio no está en una versión estable de su desarrollo actualmente.

4.2 Versión del software

Al desarrollar una aplicación en Android, debemos seleccionar cual es la mínima versión del sistema operativo Android que dicha aplicación soportará; es decir, en que dispositivos móviles será capaz de ejecutarse nuestra aplicación. Cuanto mayor sea la versión mínima requerida, menos dispositivos serán capaces de ejecutarla, pues muchos de ellos no dispondrán de la versión del sistema operativo más nueva. Por otro lado cuanto menor sea esta versión mínima requerida para ejecutar la aplicación, de menor funcionalidad dispondremos para implementar en la misma.

Debemos decidir si preferimos una aplicación que se adapte a cualquier dispositivo pero que tenga una funcionalidad bastante limitada o, de lo contrario, limitar el acceso y la compatibilidad con la aplicación, consiguiendo a cambio una mayor funcionalidad y libertad a la hora de desarrollar. Normalmente, se suelen desarrollar las aplicaciones para las últimas versiones del sistema operativo, aprovechando las prestaciones de este; pero en este caso, siendo una aplicación accesible, y como no todo el mundo puede permitirse un dispositivo móvil de última generación, he decidido utilizar la funcionalidad necesaria para la aplicación sin restringir excesivamente su uso. Por ello, estableceremos la versión mínima del sistema operativo lo más bajo posible, sin renunciar a la funcionalidad necesaria para implementar todos los requisitos de la aplicación.

Esta versión es la Android 4.0 Ice_Cream Sandwich (API Level 14). Esta versión es la versión más reciente que implemento funcionalidad dentro del sistema operativo que es necesaria para el desarrollo de nuestra aplicación. Las novedades que incluye esta versión frente a su antecesora que nos interesan para el desarrollo del proyecto son:

- **Calendar provider.** Esta nueva API nos permite tratar con información relativa a fechas y horas. Esta API nos será de gran utilidad al trabajar por ejemplo con las horas de conexión de los contactos, las horas de envío de mensajes, etc...

- **Explore-by-touch.** Nos permitirá implementar funcionalidad muy útil para el desarrollo de la accesibilidad para personas con deficiencia visual. Esta funcionalidad nos da la posibilidad de describir lo que hay bajo nuestro dedo cuando pasamos el mismo por la pantalla.

- **Text-to-Speech Engine.** Al igual que el punto anterior, esta nueva actualización incluye aspectos relacionados con la accesibilidad para personas con deficiencia visual. En este caso, la funcionalidad de la siguiente API consiste en convertir en audio cualquier texto que nosotros definamos, haciendo posible la lectura de los mensajes de una conversación, entre otras cosas.

- **Popup Menus.** La capacidad de lanzar ventanas con diferentes opciones interactuables ya fue implementada en la versión 3.0 de Android pero en esta versión son esta funcionalidad es actualizada otorgándole a la API mayores posibilidades.

Además de estas APIs mencionadas esta versión de Android incluye otras mejoras en el sistema operativo, pero dichas mejoras no suponen un beneficio a nuestro proyecto, pues son innecesarias.

4.3 Servidor web

4.3.1 Análisis de lenguajes

En toda aplicación en la que se contacte con otros dispositivos utilizando internet es necesario el uso de un servidor, que haga de medio entre estos dispositivos, así nosotros enviaremos información al servidor desde nuestro dispositivo y otros dispositivos podrán consultar dicha información en el mismo. Es decir, este servidor almacenará en una base de datos toda la información que sea necesario compartir con otros dispositivos.

Para hacer posible la comunicación entre el dispositivo y la base de datos global es necesario crear un servidor en un determinado lenguaje de programación web. En este caso evaluaremos distintas opciones y elegiremos la que mejor se adapte a nuestras necesidades para desarrollar el servidor. Estas opciones son ASP.NET, Java -JSP y PHP.

- **ASP.NET:** Este framework para aplicaciones web se caracteriza principalmente por ser muy ligero y por su facilidad para crear estas aplicaciones, permitiendo crear grandes páginas con mucho menos código que otros lenguajes.
  Compila el código de forma dinámica y posee una gran cantidad de clases que nos brindarán una gran maniobrabilidad al desarrollar la funcionalidad de la aplicación web.
  Una de las desventajas de este framework reside en el desarrollo de scripts, pues estos resultan bastante más complejos de implementar. Sin embargo,

como nuestra aplicación web sólo será un medio de comunicación entre el dispositivo y la base de datos, este punto no nos afecta en absoluto.

- **Java-JSP:** Java es un lenguaje muy sencillo de utilizar, que posee una curva de aprendizaje muy rápida. Al igual que .NET la interpretación y compilación del lenguaje es dinámica, ralentizando la ejecución de la aplicación lo menos posible.
  Es distribuido, lo que quiere decir que Java nos ofrece una gran colección de clases que nos serán de gran utilidad en el uso de aplicaciones web.
  Una de las mejores características de Java es que es un lenguaje orientado a objetos. Un lenguaje estructurado en grupos de datos que facilitarán mucho la manipulación de los mismos.
  Para comunicarnos con la base de datos utilizando Java, lo más sencillo sería utilizar JDBC, una API de Java que nos permitirá realizar operaciones en la base de datos desde el propio código Java.

- **PHP:** PHP es un lenguaje libre lo que le convierte en un lenguaje de programación muy utilizado que dispone de una gran cantidad de características y una documentación clara a la que es muy sencillo acceder.
  PHP posee soporte con numerosas bases de datos entre las que están: MySQL, Oracle, MS SQL Server... característica muy importante para nosotros, pues la principal finalidad de la aplicación web es la comunicación con la base de datos servidor.
  Soporta en cierta medida, al igual que Java, la orientación a objetos, y se caracteriza por ser un lenguaje rápido en ejecución y sencillo de aprender.
  No necesita que se definan los tipos de variables ni el tratamiento del código a bajo nivel, lo que lo convierte en un lenguaje más intuitivo y más rápido mientras desarrollamos.
  Para poder probar nuestra aplicación web será necesario instalarla en un servidor web, lo que dificulta y ralentiza el testeo de la misma. Sin embargo, podemos instalar un servidor web local en nuestro equipo para probar dicho código PHP desde el, sin necesidad de instalarlo en el servidor cada vez que realizamos algún cambio.

Tras el análisis de estos tres lenguajes de programación, decidí que la mejor opción era desarrollar esta aplicación web en **PHP**. Aunque .NET y Java son lenguajes de programación muy potentes, PHP nos ofrece ciertas características que nos ayudarán mucho más en el desarrollo de nuestra aplicación web.

Uno de las ventajas que PHP nos da es la versatilidad y facilidad de conexión con numerosos tipos de bases de datos, lo que es muy importante en nuestro servidor,

pues este tendrá dos funciones básicas: conectarse con la base de datos servidor y conectarse con el dispositivo móvil.

Aunque Java es un lenguaje orientado a objetos completamente y es una característica que nos vendría bien en el desarrollo de la aplicación, es un punto del que podremos prescindir, pues realmente nuestro servidor es un período de transición de los datos, desde el dispositivo móvil a la base de datos y viceversa; por ello, el hecho de que estos datos estén organizados y estructurados como objetos, no nos da una gran ventaja sobre el desarrollo. Además PHP no nos exige la definición del tipo de datos que utilizamos, lo que nos facilitara la manipulación de estos, pues podremos recibirlos y enviarlos sin preocuparnos del tipo que son y tratando todos los datos manipulados de la misma forma.

Uno de los problemas de la utilización de PHP para el desarrollo del servidor son las complicaciones para el testeo del mismo. Sin embargo, este problema puede quedar solucionado desde un principio, instalando un entorno de desarrollo con un servidor web local, que nos permita probar todo el código antes de instalarlo en el propio servidor.

4.3.2 Instalación de entorno de trabajo

Para el desarrollo del servidor web utilizaremos dos herramientas principalmente, una para desarrollar y escribir nuestro código PHP y otra para ayudarnos a probar el mismo.

Para la escritura de nuestro código en PHP podremos utilizar cualquier procesador de textos o un IDE especializado para este fin. Personalmente recomiendo utilizar un IDE, especialmente si no tenemos experiencia previa programando en PHP, ya que nos ayudará corrigiendo sintácticamente y señalando nuestros errores simultáneamente a su escritura. Los IDEs más utilizados para estos fines, como al desarrollar en Java, son Eclipse y Netbeans, por otro lado, es necesario instalar plugins para su habilitación para el desarrollo de PHP.

En este caso, la elección del IDE no es realmente importante, ambos nos ofrecerán similares funcionalidades. Dado que no estoy acostumbrado a trabajar con Netbeans y prefería no mezclar ambos proyectos (aplicación cliente y servidor) en el IDE de Eclipse, opté por buscar una alternativa a estos. Encontré Aptana Studio 3, un IDE centrado completamente en PHP y programado sobre Eclipse, por estas razones decidí utilizarlo para ayudarme en el desarrollo del servidor, ya que estaba acostumbrado a trabajar con Eclipse y este poseía una interfaz idéntica.

Para hacer más sencillas las pruebas de la aplicación nos ayudaremos de XAMPP. Este servidor nos ayudará a implementar 3 herramientas que nos serán de gran utilidad al desarrollar nuestro servidor web: un intérprete de PHP, MySQL y un servidor web Apache.

Esta herramienta nos permitirá ejecutar nuestro código PHP en un servidor local desplegado en nuestro equipo gracias a Apache. De este modo, podremos realizar todas las pruebas convenientes sobre nuestro código desde nuestro navegador; sin necesidad de instalar nuestro proyecto en un servidor externo que necesitaríamos en caso de no disponer de Apache.

4.4 Almacenamiento de datos. Aplicación cliente

Para almacenar datos de forma permanente dentro de la aplicación cliente de nuestro proyecto, tendremos varias opciones:

Podemos hacer uso del almacenamiento en ficheros externos a la aplicación; ficheros almacenados en el propio dispositivo. Estos ficheros podrían ser tanto ficheros de texto normales como ficheros en formato XML. Esta opción ha sido descartada desde un primer momento, ya que los datos que pretendemos almacenar en la aplicación deben ser leídos por la misma aplicación evitando en la medida de lo posible el acceso externo a estos datos por cualquier usuario, aumentando así la seguridad del sistema.

Llegados a esta conclusión tendremos dos maneras alternativas de almacenar permanentemente la información de nuestra aplicación, que serán utilizadas para fines diferentes.

En primer lugar tenemos el almacenamiento por medio de preferencias. Cada una de estas preferencias será una variable que se almacenará en la aplicación, pudiendo acceder a la misma en cualquier momento de su ejecución. Este tipo de almacenamiento será utilizado para guardar las preferencias de un usuario, recordar el último usuario que hizo *login* en la aplicación... En general, guardar datos únicos que no tenga una cantidad de datos variable.

Con el resto de información, la cual no podremos conocer su tamaño, como por ejemplo, los contactos o las conversaciones de un usuario; utilizaremos una base de datos, donde almacenaremos este tipo de información. Desarrollando una aplicación Android solo tendremos acceso al uso de un tipo de base de datos, esta es SQLite. Este sistema de bases de datos relacional se integra completamente con nuestro programa

haciendo posible que, de forma sencilla y sin tener que escribir mucho código, podamos manipular la base de datos de nuestra aplicación.

4.5 Base de datos servidor

Para guardar toda la información de la aplicación y poder compartirla con los diferentes dispositivos que hacen uso de la misma, será necesario implementar un servidor web conectado a una base de datos. Esta base de datos estará basada en SQL aunque para su implementación podrán ser utilizados varios motores de bases de datos y, al haber escogido PHP como nuestro lenguaje de desarrollo del servidor, contamos con la ventaja de poder utilizar la mayor parte de estos motores de bases de datos.

Entre los más importantes y más utilizados están Oracle, MySQL y SQL Server; que son los que, a continuación, analizaremos para elegir el que mejor se adapte a las necesidades de nuestra base de datos y al desarrollo y acceso de la misma.

Oracle es un motor de bases de datos realmente robusto y seguro lo que nos sería de gran utilidad en nuestra aplicación, ya que tratamos con información de usuarios privada que no debería filtrarse al exterior por la propia seguridad del usuario.
Por otro lado, Oracle es uno de los motores de bases de datos más complejo de establecer y utilizar; además, la mayoría de las versiones de este motor no so de libre uso y tienen un coste asociado a su utilización.

SQL Server es otro de los motores de bases de datos más utilizados. Al igual que Oracle, la mayoría de sus versiones de pago, con lo que dificultaría su utilización en el proyecto. Además, este motor no te permite la posibilidad de elegir diferentes formas de almacenamiento.

MySQL es un motor de código abierto y uso gratuito, lo que nos concedería más libertad de prueba sobre el mismo motor. La interfaz del motor es simple y por medio del terminal, lo que haría algo más complicada la manipulación de los datos de la base de datos. Por otro lado, existen plugins y herramientas para hacer esta tarea mucho más sencilla.
Trabajando en local y habiendo instalado como se explicó anteriormente la herramienta XAMPP, podremos acceder a una interfaz (phpMyAdmin) para administrar y configurar nuestra base de datos MySQL de forma rápida y sencilla.

Una vez analizados los motores de bases de datos, decidí que la mejor opción era utilizar MySQL ya que es un motor de uso gratuito y se adaptada perfectamente a las necesidades de la aplicación y al entorno de trabajo que habíamos instalado con anterioridad.

4.6 Alojamiento del servidor

Si queremos desarrollar una aplicación web será necesario que poseamos un servidor con un dominio asociado. En nuestro caso, encontré una página web que ofrecía servidores web de forma totalmente gratuita (hostinger.es).

Con tan solo registrarnos en dicha página web, podremos crear nuestro propio servidor web. Este servidor estará asociado a nuestra cuenta de usuario de la página y podremos acceder a él desde cualquier lugar. Desde la interfaz de la página podremos tener a nuestra disposición numerosas funcionalidades que nos harán mucho más sencilla la tarea de desplegar nuestro proyecto en el servidor web.

De estas funcionalidades, para nuestras necesidades, podremos destacar un administrador de archivos, donde cargaremos nuestra aplicación web copiándola desde nuestro equipo. Tras realizar esta simple tarea nuestro servidor estará activo y podrá ser utilizado desde cualquier dispositivo que lo requiera.

Esta página web nos ofrece la posibilidad de crear una base de datos MySQL en el servidor asociado a nuestro dominio, por ello, utilizar MySQL como motor de base de datos para el servidor web nos facilitará muchos las cosas. Desde la página, dispondremos de una interfaz igual que phpMyAdmin que nos ofrece XAMPP en nuestra base de datos local de pruebas, con lo que trasladar y manipular esta base de datos en el servidor no tendrá ninguna diferencia con hacerlo en nuestro equipo.

4.7 Google Cloud Messaging

Mediante la aplicación cliente y la aplicación servidor de nuestro proyecto, cualquier dispositivo será capaz de comunicarse con el servidor enviando una petición y este, podrá recibir información en forma de respuesta. Por otro lado, el servidor no será capaz de comunicarse con un dispositivo en el momento que desee, ya que no tendrá forma de identificarlos, es por esta razón que, en el caso de tener que enviar un mensaje de un dispositivo a otro, no sería posible contactar con el segundo dispositivo.

Deberíamos esperar a que este segundo dispositivo se conectara con el servidor, "preguntando" si hay mensajes pendientes para dicho dispositivo.

Esta sería una solución al problema, mandar una petición al servidor cada vez que accedamos a la aplicación para saber si hay algún mensaje o actualización en la base de datos, que el dispositivo deba conocer. Esto nos plantea el problema o incomodidad de no poder saber si hay mensajes nuevos en la aplicación sin abrir la misma.

Por esta razón, utilizaremos un servicio de Google especialmente desarrollado para este fin: Google Cloud Messaging.

GCM es un servicio de Google que nos permitirá enviar notificaciones a cualquier dispositivo registrado en su sistema realizando una petición desde nuestro servidor. Funciona de la siguiente manera (Figura 6):

1. Al registrarnos en nuestra aplicación, además de incluir nuestro usuario en la base de datos del servidor. Mandaremos una petición a este servicio de Google.
2. GCM nos registrará en el sistema y nos devolverá una clave del dispositivo. Esta clave será la que facilitaremos en un futuro al servicio GCM para que el sepa a qué dispositivo tiene que enviar la notificación.
3. Enviaremos esta clave a la base de datos del servidor para que sea almacenada asociándola con nuestro usuario.
4. Al mandar un mensaje, se establecerá una conexión con el servidor con el usuario al que se quiere enviar el mensaje, el mensaje en cuestión e información relativa al mensaje.
5. El servidor recibirá esta petición y, aparte de almacenar toda esta información recibida en la base de datos, enviará una petición a GCM junto con los datos del mensaje necesarios y la clave del receptor del mensaje. Además, tendrá que enviar una clave de uso del servicio asociada a nuestro proyecto en Google Developers por motivos de seguridad para que, solo los que faciliten dicha clave, sean capaces de enviar notificaciones.
6. El servicio GCM recibirá la información adjunta a la petición del web service y mandará una notificación a todos las claves de receptor facilitadas en la petición.
7. Sin necesidad de tener abierta la aplicación y gracias a la clase GCMIntentService, la aplicación cliente será capaz de recibir cualquier notificación y administrarla en el sistema. Esta clase será un servicio y no una actividad, lo que implica que no será necesario que la aplicación se esté ejecutando para que se ejecute el código de la clase.

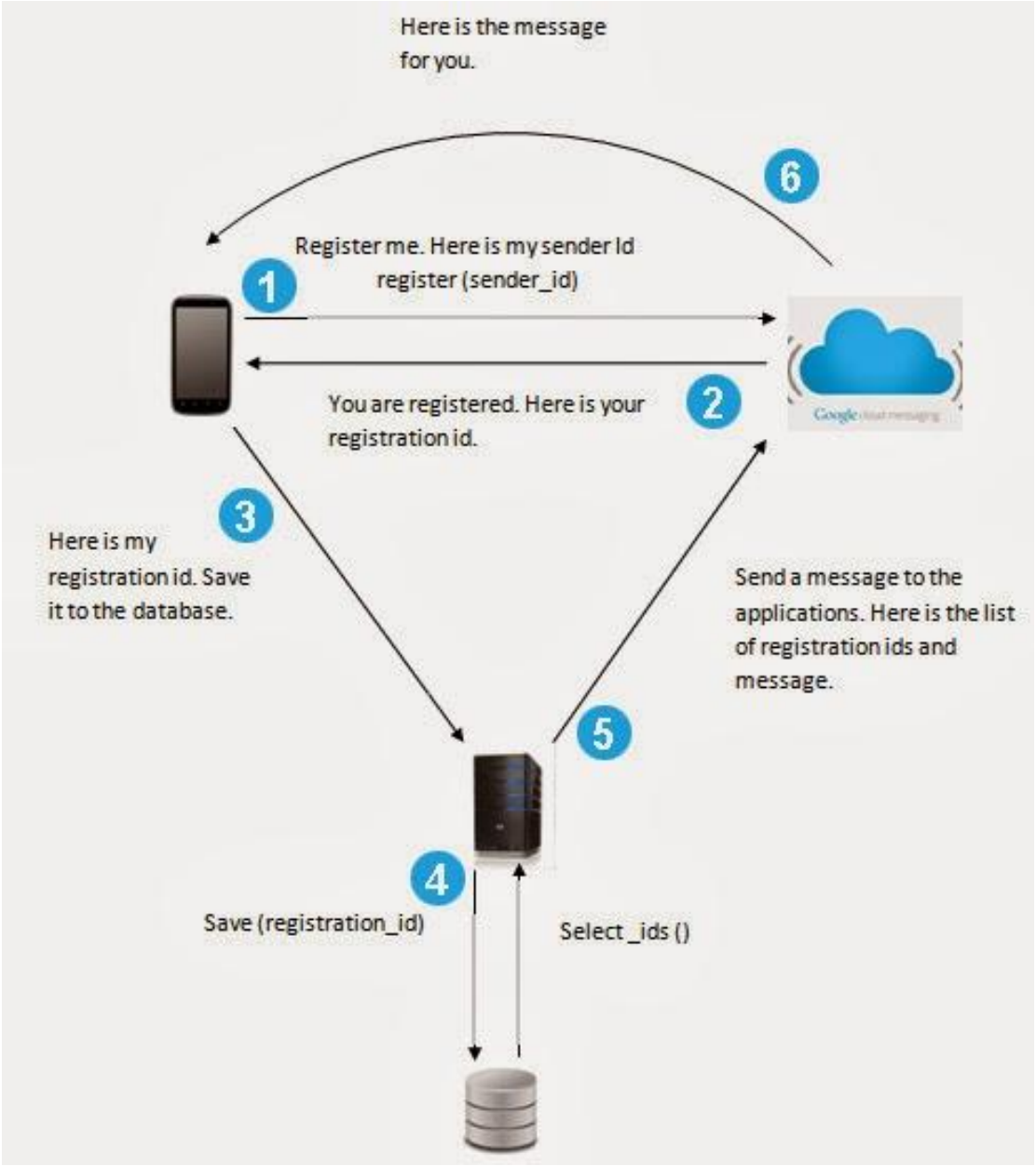

Figura 6: Esquema de comunicación con el servicio GCM

Capítulo 5
Desarrollo de la aplicación

Durante todo el desarrollo de la aplicación y en cada fase del mismo, se proporcionaron guías de desarrollo, guías que determinan como debe funcionar la fase especifica a la que la guía se refiere. Cada una de estas guías tiene ciertos requisitos y explicaciones que se han tenido que integrar en la aplicación. Junto con este documento se adjunta una de estas guías de desarrollo como ejemplo para mostrar el punto del que se parte para el desarrollo de las fases.

Ahora, se procederá a la explicación de cada una de las fases, explicando su estructura, cómo se han programado y cómo se deberían utilizar dentro de la aplicación.

5.1 Almacenamiento de datos

Para almacenar los datos de la aplicación utilizaremos tres métodos distintos, y para cada tipo de dato escogeremos el método de almacenamiento que se adapte al mismo.

Manipularemos información relativa a las preferencias del usuario que no es necesaria compartir con otros usuarios y es información de tamaño no variable, es decir, datos únicos que definirán las preferencias. Para esta información utilizaremos la clase SharedPreferences, que nos permite acceder a sus variables y modificarlas en cualquier punto de la ejecución del programa.

Para almacenar información de la que no conocemos su tamaño, dentro de la aplicación de cada usuario (conversaciones, contactos...) utilizaremos una base de datos SQLite.

A continuación explicaré el diagrama de clases de la base de datos de la aplicación cliente, así como todas sus tablas y relaciones:

- **Conversation.** En esta tabla guardaremos toda la información relativa a las conversaciones en las que participa el usuario. Esta información comprende:

    - **id.** Identificador de la conversación. Clave primaria.

- **previousMsgAccess.** Valor que nos indicará si, en determinada conversación, es posible ver mensajes anteriores a la fecha de nuestra inclusión en dicha conversación.
- **name.** Nombre de la conversación.
- **type.** Tipo de la conversación. Determinará si la conversación es grupal, o de sólo dos usuarios.
- **lastMsg.** Almacenará el último mensaje de dicha conversación. Esta información ya es almacenada en los mensajes, pues guardamos la fecha de envío, pero será útil para no tener que acceder a los mensajes y obtener este dato mucho más rápido al mostrarlo en la selección de conversaciones.
- **dateLastMsg.** Almacenará la fecha del último mensaje de la conversación. La razón es la misma que en el campo anterior.
- **block.** Este valor definirá si una conversación está bloqueada por el usuario o no. En caso de que lo este, no será posible recibir ni enviar mensajes en la misma.
- **date.** En el caso de que sea un *Meeting*, es decir, sea un debate con fecha y tema concertados, este valor será igual a la fecha en la que dicho debate empezará.
- **topic.** En las mismas condiciones que el campo anterior, este determinará el tema que será tratado durante la conversación.
- **terms.** Propiedad en la que se guardarán los términos a los que se refiere la conversación, motivos de esta. Almacenará las palabras o temas que se tratarán en la conversación. Define y especifica el *topic*.
- **imageDate.** Se almacenará la fecha del último cambio de imagen. De esta forma, todas las aplicaciones cliente podrán comprobar si la imagen del usuario o conversación han sido cambiadas, sin necesidad de descargar la imagen cada vez; sólo consultando esta fecha.
- **image.** Guarda la imagen de la conversación. En caso de ser una conversación de dos únicos usuarios, su valor será nulo.

- **Contact:** En esta tabla guardaremos todos los datos de los contactos de nuestro usuario. Además, guardaremos las preferencias especificas de cada uno de estos, como por ejemplo, la fuente de letra en la que queremos que se muestren sus mensajes.

  - **id.** El identificador del contacto. Clave primaria.
  - **image.** Imagen del contacto.
  - **state.** Estado del contacto. Este estado es personalizable para cada usuario.
  - **nickname.** Nombre de usuario que se mostrará en las conversaciones. Al principio, este será igual al *nickname* que el usuario en cuestión defina. Sin

embargo, una vez añadido a tus contactos, cada usuario será capaz de modificar los *nickname* de sus contactos para su dispositivo.
- **block.** Este valor determinará si uno de tus contactos está bloqueado. En el caso de que lo esté, no será posible tanto enviar como recibir mensajes de dicho usuario.
- **lastConnection.** Almacena la última conexión a la aplicación de el contacto en cuestión.
- **fontType.** Define el tipo de fuente en el cual se mostrarán los mensajes del contacto.
- **imageDate.** Almacena la fecha del último cambio de imagen del usuario. Al igual que en las imágenes de cada conversación, este dato se utilizará para verificar si un usuario a modificado su imagen y, en ese caso, descargar la nueva imagen.

- **Contact-Conv:** Esta clase es producto de una relación *"muchos a muchos"* entre Contact y Conversation. Para relacionar cada usuario con todas las conversaciones a las que pertenece, así como todas las conversaciones con todos los usuarios que participen en ella, es necesaria una tabla intermedia. Además, utilizaremos esta tabla para guardar la información de un determinado usuario en una determinada conversación.

  - **idconver.** Clave foránea que apunta a la clave primaria de Conversation. Identifica la conversación. Forma la clave primaria de la tabla junto con *"user"*.
  - **user.** Clave foránea que apunta a la clave primaria de Contact. Identifica al usuario al que se refiere la tupla. La clave primaria está formada por **idconver** y **user**, asegurándonos que cada tupla identifica a un determinado contacto en una determinada conversación.
  - **leader.** Valor que determina si un usuario en una conversación tiene permisos de líder (agregar otros contactos, cambiar nombre, cambiar imagen...). Un usuario puede ser líder en unas conversaciones y puede no serlo en otras.

- **Message:** En esta tabla se guardará toda la información relativa a los mensajes.

  - **id.** Identificador del mensaje. Será de utilidad para filtrar y seleccionar el mensaje que necesitemos. Clave primaria.
  - **datetime.** Fecha y hora de envío del mensaje. Para evitar problemas de zonas horarias y que los mensajes no se muestren en orden correcto por este motivo, todas las fechas serán generadas en el servidor.

- **type.** Tipo del mensaje. Este tipo puede tener tres diferentes valores, que determinarán si el mensaje es un texto, un enlace o un archivo adjunto. Dichos tipos de mensaje se definirán posteriormente.
- **state.** Estado del mensaje. En el caso de que el mensaje lo hayamos enviado nosotros, determinará si el mensaje ha sido recibido por el receptor o tan solo enviado por nosotros(En el caso de una conversación de dos usuarios). Si el mensaje ha sido enviado por otro usuario a nosotros, el estado le indicará a la aplicación que el mensaje no ha sido visto y debe resaltarlo de alguna forma (notificación, aviso en pantalla de selección de conversaciones...).
- **groupImp.** Este valor indicará si el mensaje está marcado como importante para todo el grupo dentro de la conversación. Este valor solo podrá ser modificado por los líderes del grupo.
- **user.** Determina el usuario que envió el mensaje. Clave foránea de la clave primaria de Contact.
- **idconv.** Determina a que conversación pertenece el mensaje. Clave foránea de Conversation.
- **important.** Determina si un mensaje es importante para el usuario que manipula la aplicación. Este valor se podrá cambiar únicamente por y para el usuario en la aplicación cliente, no será visible para el resto de contactos de la conversación. Sirve únicamente de marcación personal.

- **Accessibility:** Esta tabla define lo accesible que es el contenido de un enlace o un archivo adjunto. Establece una relación *"uno a muchos"* con mensaje, pues un mensaje puede tener varias condiciones de accesibilidad. Es una característica única de los enlaces y los archivos adjuntos, sin embargo, como la clave primaria de estos coincide con la del mensaje al que pertenecen y por no crear dos tablas iguales para un mismo fin, esta tabla se relaciona con Message; y no con Link y Attached.

  - **id.** Determina el identificador del elemento de accesibilidad. Clave primaria.
  - **idMsg.** Indica el mensaje al cual pertenece el elemento de accesibilidad. Clave foránea de Message.
  - **accessibility.** Define la condición de accesibilidad a la que está ligado el mensaje.

- **Link:** Tabla que define un tipo de mensaje. Ya que para cada tipo de mensaje, hay diferentes valores que almacenar, es necesario crear tablas diferentes. Al ser una relación directa *"uno a uno"* con Message, la clave primaria puede ser la misma que en Message; y como necesariamente un enlace tiene que estar

ligado a un mensaje, esta clave primaria será, además, clave foránea de Message.

- **idMsg.** Determina el mensaje al que pertenece el enlace. Además, nos sirve para identificar el enlace dentro de la tabla. Clave primaria y clave foránea de Message.
- **link.** Almacena el propio enlace.
- **description.** Este valor describe el contenido del enlace y lo que podemos encontrar tras este.
- **language.** Guarda el idioma en el que está el contenido al que te lleva el enlace. Valor relacionado con la accesibilidad del mensaje.

- **Attached:** Tabla que define las características de un mensaje de tipo archivo adjunto. La clave primaria del mismo, comprende las mismas características que el de la tabla Link.

  - **idMsg.** Determina el mensaje al que pertenece el enlace. Además, nos sirve para identificar el enlace dentro de la tabla. Clave primaria y clave foránea de Message.
  - **language.** Almacena el lenguaje en el que se trata la información que presenta el archivo adjunto (en caso de un archivo de texto, el idioma en el que está escrito, en caso de un archivo multimedia de audio o video, el idioma en el que se hable en dicho archivo...).
  - **size.** Tamaño del archivo adjunto que se envía junto al mensaje.
  - **name.** Nombre que se le da al archivo.

- **TextMessage:** Tabla en la que se guarda el tercer tipo de mensajes. Su clave foránea también comparte las características de los otros dos tipos de mensajes.

  - **idMsg.** Determina el mensaje al que pertenece el enlace. Además, nos sirve para identificar el enlace dentro de la tabla. Clave primaria y clave foránea de Message.
  - **text.** Almacena el texto enviado con el mensaje.

A continuación podemos ver el diagrama de clases que representa la estructura de la base de datos anteriormente explicada (Figura 7).

Las líneas continuas representan relaciones de clave foránea que a su vez son clave primaria en su propia tabla.
Las líneas discontinuas representan claves foráneas a otras tablas.

El icono de cada llave amarilla en los atributos de las tablas representa una clave primaria, el rombo de color azul representa un atributo obligatorio, el rombo de color rojo representa una clave foránea que depende de la primaria de otra tabla y el rombo blanco representa un atributo de la tabla no obligatorio.

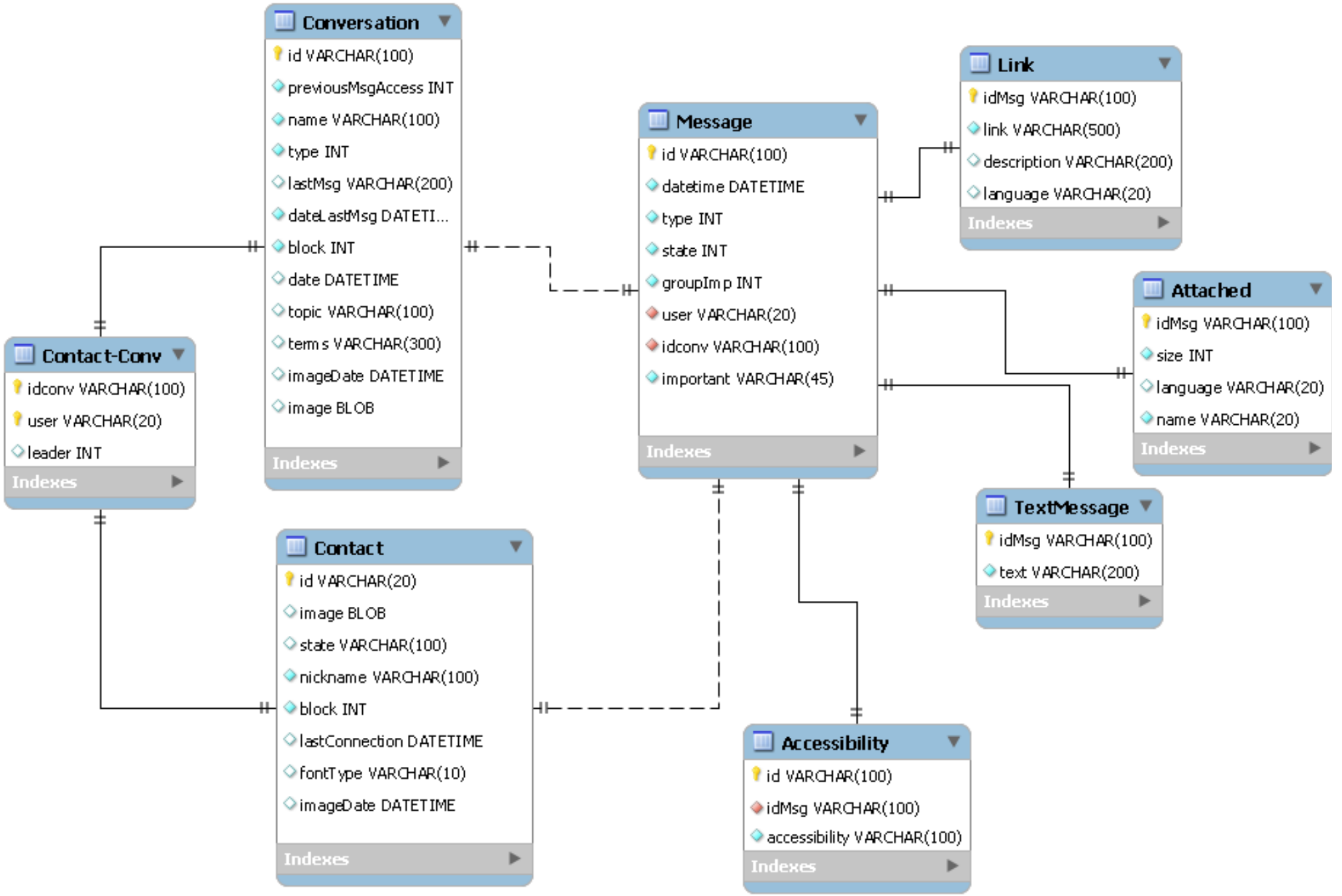

Figura 7: Diagrama de clases de la base de datos cliente

Tras analizar la base de datos de la aplicación cliente, el último tipo de datos que debemos almacenar los datos globales, datos que deberemos compartir con todos los usuarios o que serán necesarios para realizar diferentes funciones en el servidor.

Esta base de datos será en MySQL y accederemos a ella desde nuestro servidor PHP. Gran parte de esta base de datos será idéntica a la base de datos de la aplicación cliente; sin embargo, esta almacenará la información de todos los usuarios y no solo de uno de estos, pues esta será utilizada por todos los dispositivos que tengan la aplicación instalada.

Ya que comparte mucha información con idéntico significado que la base de datos cliente, me limitaré a explicar las diferencias entre ambas:

- **Message.** En el servidor no almacenaremos el valor *"important"* dado que dicho valor solo indica si un mensaje es importante o no individualmente, información diferente para cada usuario. Al contrario pasa con *"groupImp"*, que es un dato global, igual para todos los propietarios de dicho mensaje.

- **Contact.** En esta tabla desechamos un atributo que tenemos en la aplicación cliente:

  - **fontType.** Este dato es innecesario en el servidor, ya que es un dato que solamente regulará la forma de visualizar los datos en la aplicación cliente.

  Además, añadiremos ciertos atributos que nos servirán para identificar a los usuarios de la aplicación:

  - **receiverId.** Identificador de GCM. Será utilizado para determinar a qué dispositivo mandar el mensaje deseado. Cada usuario tiene su propio *"receiverId"* para identificarlo dentro del servicio GCM.
  - **pass.** Contraseña utilizada para limitar el acceso a la aplicación. No será posible conectarse como un usuario si no se conoce su contraseña asociada.
  - **question.** Valor asociado con la seguridad de las cuentas de usuario. Si un usuario de la aplicación olvidara su contraseña, podría recuperarla en el caso de que recuerde esta pregunta y la correspondiente respuesta.
  - **answer.** Respuesta a la pregunta de seguridad. Utilizada para verificar la autenticidad de un usuario al solicitar su contraseña en caso de no recordarla.

El resto de información almacenada en la base de datos servidor es igual a la información guardada en la aplicación cliente. A continuación podemos ver su diagrama de clases correspondiente (Figura 8):

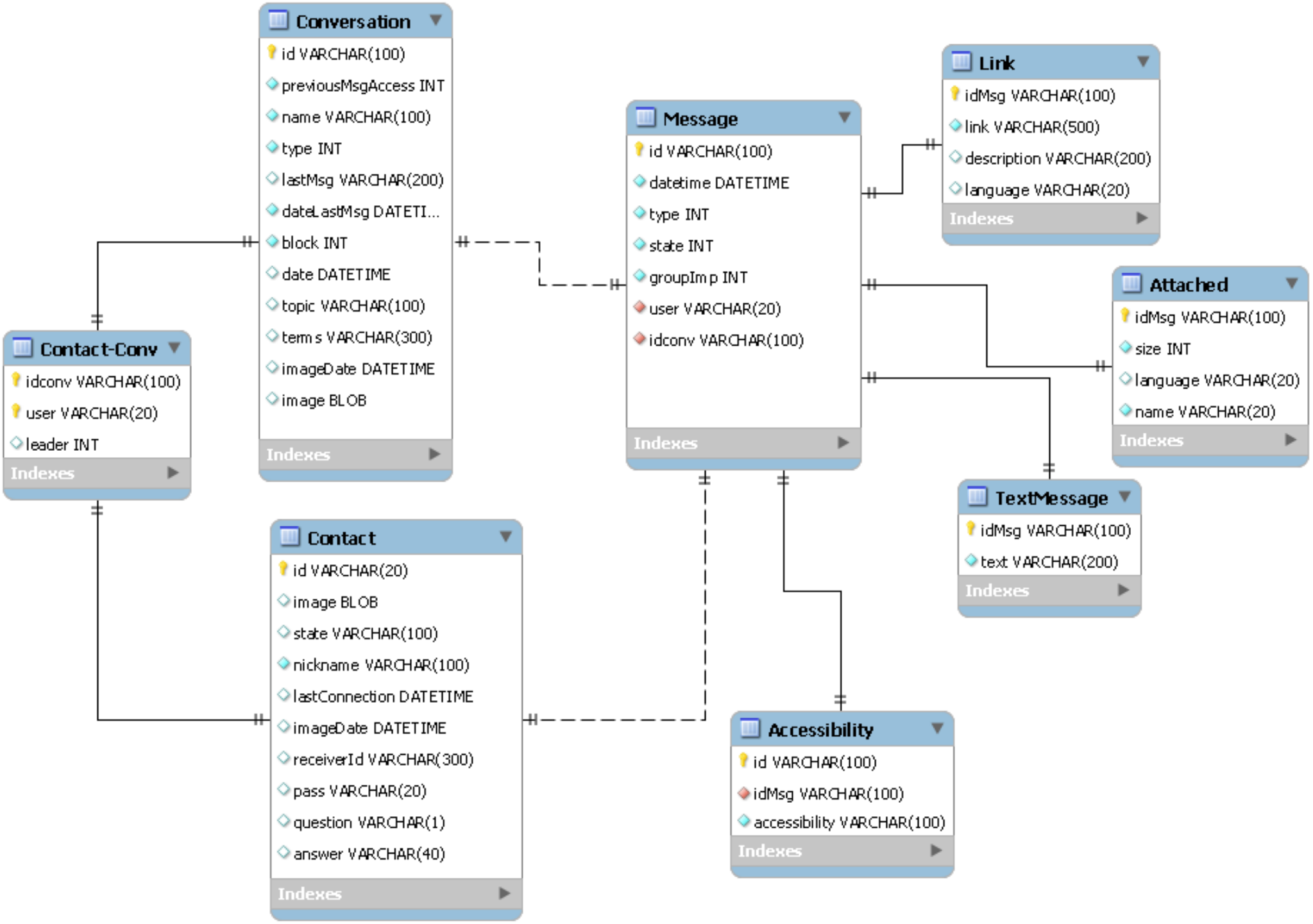

Figura 8: Diagrama de clases de la base de datos servidor

Para enviar información desde la aplicación cliente al servidor será necesario crear una conexión entre ambos. Para ello utilizaremos el protocolo HTTP, con el cual enviaremos desde nuestra aplicación Android peticiones al servidor con argumentos enviados bajo el método POST (Figura 9).

Figura 9: Comunicación cliente-servidor

5.2 Login y preferencias de usuario

En esta fase implementaremos la interfaz y funcionalidad de las pantallas de acceso a la aplicación (Login) y de las preferencias de usuario.

5.2.1 Login

En la parte gráfica de esta pantalla dispondremos de dos campos de texto para rellenar: usuario y contraseña. También tendremos un interruptor que nos permitirá mostrar y ocultar los caracteres del campo de la contraseña (Figura 10).

Finalmente en la parte baja de la pantalla dispondremos de dos botones de acceso a la aplicación y registro de usuario, respectivamente; seguidos de un texto seleccionable

"Olvidaste la contraseña?" que nos conducirá a una pantalla donde recuperar nuestra contraseña en el caso de que dispongamos de una cuenta de usuario y no recordemos los datos de acceso.

Para acceder a la aplicación será necesario introducir un usuario y una contraseña que coincidan con los datos de nuestra base de datos en el servidor, para así, poder verificar el propietario de la cuenta. Para crear una cuenta de usuario únicamente será necesario pulsar el botón adecuado para ser reconducidos a otra página donde podremos introducir los datos de nuestra nueva cuenta de usuario.

Además, en la parte lógica y no visible de esta página, controlaremos el recuerdo del último usuario, así como un *login* automático del mismo.

En la página de creación de cuentas de usuario (Figura 11) dispondremos de varios campos de escritura: usuario, contraseña, repetición de contraseña, pregunta de seguridad y respuesta; y un botón para proceder con la creación de la cuenta una vez rellenados los campos. En esta página se comprobará que el usuario no está en uso en la aplicación, la correcta escritura de todos los campos, que las contraseñas proporcionadas sean iguales... Filtros que ayudarán al correcto funcionamiento de la aplicación.

Una vez presionado el botón de acceso a la aplicación y que la información de acceso coincida con la de un usuario existente, o cuando creemos nuestra cuenta de usuario con éxito; accederemos a la pantalla principal de la aplicación donde veremos las conversaciones activas que tenemos.

Figura 10: Pantalla de login

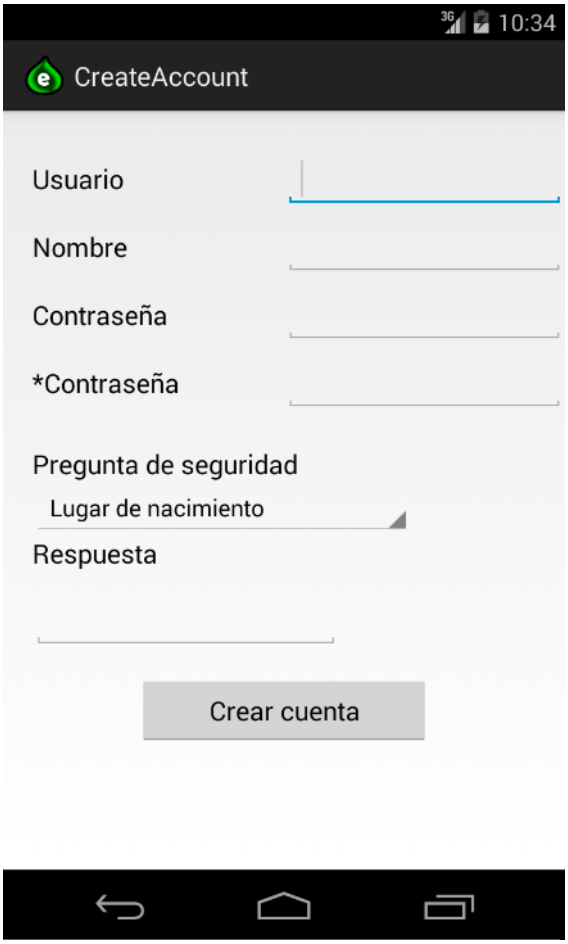

Figura 11: Pantalla de creación de cuenta de usuario

5.2.2 Preferencias de usuario

Desde la pantalla principal de la aplicación donde vemos todas las conversaciones activas que tenemos podremos acceder desde el menú arriba a la derecha de la misma, a la pantalla de preferencias de usuario.

En esta pantalla podremos personalizar como es mostrada o emitida la información de la aplicación. Constará, en mayor medida, de una serie de cajitas marcables y lista desplegables que nos ayudarán a adaptar la aplicación a nuestras necesidades. Por otro lado, en esta pantalla solamente se guardarán las preferencias en el dispositivo, con la clase *SharedPreferences* y, cuando sea necesario mostrar algo, o emitir una notificación de una determinada manera especificada en estas preferencias, esta clase será consultada; para decidir, en función de lo seleccionado por el usuario, de qué manera mostrar la información (Figura 12).

Los valores y preferencias que podremos personalizar desde esta página son los siguientes:

- **Mostrar fecha de cada mensaje**
- **Color de letra de la fecha**
- **Formato de fecha**
- **Fuente de letra de la fecha**
- **Mostrar el usuario en cada mensaje**
- **Mostrar el nombre en cada mensaje**
- **Mostrar la imagen en cada mensaje**
- **Orden de mensajes ascendente o descendente**
- **Color de letra de los mensajes**
- **Fuente de letra de los mensajes**
- **Tamaño de letra de los mensajes**
- **Número de mensajes mostrados en pantalla**
- **Bloqueo de la aplicación**
- **Tono de notificación**
- **Luz de notificación**
- **Color de la luz de notificación**
- **Notificación emergente**

Desde la misma pantalla de lista de conversaciones y mediante el mismo menú, tendremos acceso a la pantalla de personalización de nuestra información de usuario. En esta pantalla dispondremos de dos campos, para nombre y estado; y de un selector de imágenes que nos permitirá cambiar nuestra foto de perfil, tanto en nuestra base de datos como en la del servidor (Figura 13).

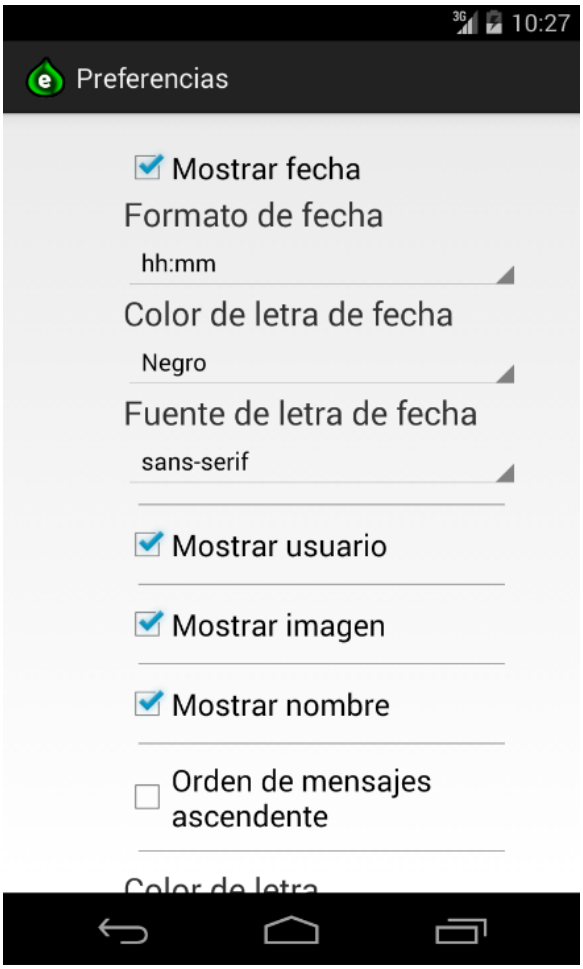

Figura 12: Pantalla de preferencias de usuario

Figura 13: Pantalla de información de usuario

5.3 Lista de conversaciones

Esta página será la página principal tras el acceso con nuestro usuario y contraseña. En ella veremos una lista de todas las conversaciones activas a las que pertenecemos y, pulsando en cada una de ellas, accederemos a la pantalla de la conversación, donde veremos los mensajes de la misma y podremos enviar nuevos.

Dispondremos también de un menú superior donde accederemos a distintas pantallas, que serán descritas cuando precise, en su determinado apartado.

Para mostrar las conversaciones haremos una consulta en nuestra base de datos local y será necesario un bucle que añada estos elementos a la interfaz gráfica de la aplicación. En la lista de conversaciones serán mostrados el nombre, la imagen, el último mensaje y la fecha del mismo de la conversación.

En esta pantalla no accederemos al servidor en busca de información, porque podría ralentizar el inicio de la aplicación y los cambios que se efectúen en las conversaciones de un usuario, serán modificados cuando este los reciba (proceso que se explicará con más detalle en el punto 5.6), por tanto podemos liberar de carga esta pantalla, aumentando su fluidez de uso.

5.4 Creación de grupos y agregar contactos

5.4.1 Creación de grupos

En esta pantalla el usuario será capaz de crear grupos y conversaciones de más de dos personas. Esta función será accesible desde el menú superior izquierdo de la pantalla principal de muestra de conversaciones.

En esta pantalla podremos especificar el nombre (obligatorio), el tema y la fecha de la conversación; y seleccionaremos los contactos de nuestra lista de contactos que queremos que sean incluidos en dicha conversación. Una vez rellenados los campos y seleccionados al menos dos participantes de la conversación podremos pulsar el botón "Crear grupo", que creará el grupo tanto en nuestra aplicación como en el servidor y nos redireccionará a la pantalla de visualización de mensajes de esta nueva conversación.

Desde la pantalla de creación de grupos no será posible añadir más información al grupo, sin embargo, desde la pantalla de la propia conversación podremos acceder a la información del mismo, para consultar, añadir o modificar nueva información.

En esta nueva pantalla podremos consultar la imagen, el nombre, la fecha, el tema y los términos más utilizados de la conversación y; en el caso de tener permisos de líder en dicho grupo, modificar estos datos. En el caso de ser líderes de la conversación podremos modificar el acceso a mensajes anteriores para nuevos usuarios de la conversación.

Dispondrá de una caja para marcar si se desea bloquear la conversación temporalmente, parando la recepción y el envío de mensajes en la misma; funcionalidad abierta para todos los usuarios de la conversación, independientemente de si son líderes o no.

Durante la creación de un grupo de conversación, el único líder será automáticamente, el creador del mismo. Sin embargo podemos cambiar esta característica, otorgando al resto de usuarios privilegios de líder de la conversación. Para acceder a esta

funcionalidad podremos hacerlo desde un botón situado en la parte inferior de la pantalla de información de la conversación.

En la nueva pantalla veremos una lista de los participantes actuales de la conversación, cada uno con un botón que nos permitirá darle a ese usuario permisos de líder, siempre y cuando, el no los tenga y nosotros sí. Bajo la lista de participantes, si tenemos permisos de líder en la conversación, tendremos un botón para acceder a nuestra actual lista de contactos, desde la cual podremos seleccionar cualquiera de ellos para añadirlo a la conversación.

5.4.2 Agregar contactos

Para iniciar una conversación con un usuario de la aplicación o añadir un usuario a una conversación ya creada será necesario que ese contacto este dentro de nuestra lista de contactos.

Para añadir un contacto a nuestra base de datos accederemos a una nueva pantalla desde la actividad de selección de conversaciones, en la parte superior de la pantalla el segundo botón a la derecha. En esta nueva pantalla dispondremos de un buscador en la parte superior, en el cual escribiremos de forma completa o parcial el usuario o el nombre de un contacto que queramos añadir (Figura 14). Seguidamente en la pantalla aparecerá una lista de usuarios de la aplicación que coincidan con la búsqueda realizada.

Una vez encontrado el usuario que queramos agregar, podremos hacerlo pulsando el botón asociado al mismo en esta pantalla.

En el caso de que un usuario que no tenemos en nuestra lista de contactos nos abra una conversación, este usuario se añadirá automáticamente a nuestra lista de contactos. E caso de que no queramos recibir mensajes de un contacto, podemos bloquearlo, al estar bloqueado, este usuario no aparecerá en nuestra lista de contactos.

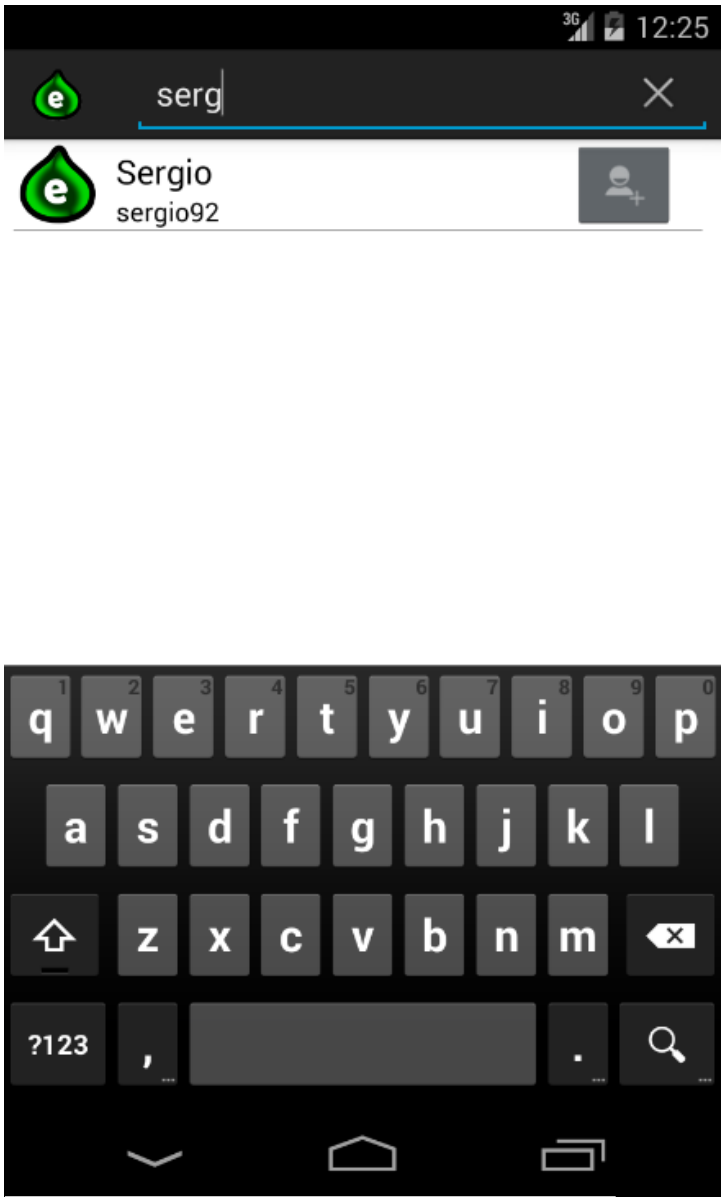

Figura 14: Pantalla para agregar contactos

5.5 Envío de mensajes

5.5.1 Conversación

La pantalla principal de esta conversación será donde se muestren todos los mensajes enviados de una determinada conversación; y estos dependerán de la conversación que hayamos seleccionado. Además, estos serán mostrados de una determinada

forma (con imágenes, nombre de usuario...) dependiendo de las preferencias que el usuario haya especificado.

Desde esta pantalla podremos acceder a la información de la conversación o la información del usuario, dependiendo si la conversación es de 2 o más personas. Por supuesto, desde esta pantalla también será posible realizar el envío de los mensajes mediante una caja de texto y un botón en la parte inferior de la pantalla, que se mantendrán fijos.

Al enviar un mensaje, la aplicación emisora realizará una petición al servidor, aclarando la conversación a la que se envía el mensaje y el mensaje en sí. Estos datos serán almacenados en el servidor, junto con la fecha y hora a la que el mensaje es generado en el servidor, que será devuelta al usuario para completar dicha información en su base de datos.

En esta pantalla, no se realizarán peticiones continuas al servidor para comprobar si hay nuevos mensajes en la conversación, pues podría ralentizar mucho la ejecución de la aplicación. Sólo se refrescarán los mensajes mostrados cada cierto tiempo desde la base de datos local y se verificará con el servidor el estado de los mensajes que aún permanezcan como "no leídos" para la actualización de este dato en caso necesario.

También se comprobará periódicamente el cambio de imagen del usuario o grupo de dicha conversación. En el caso de que la fecha de modificación no coincida con la almacenada en nuestra base de datos, se realizará una petición para la nueva imagen.

5.5.2 GCM

GCM es un servicio de Google que nos permitirá en nuestra aplicación recibir notificaciones incluso cuando esta no esté ejecutándose. Este servicio nos será de gran utilidad cuando enviemos mensajes, librándonos de la tarea de tener que consultar el servidor en busca de nuevos mensajes cada cierto tiempo.

Cuando un usuario envíe un mensaje al servidor y este lo almacene, el servidor además lanzará una petición a GCM incluyendo los identificadores GCM de cada emisor del mensaje y la información relativa al mensaje. GCM entonces se encargará de enviar esta notificación a todos los dispositivos indicados.

Para recibir dicha notificación implementaremos una clase en nuestra aplicación, que heredará de la clase *GCMBaseIntentService* de la API de Google que debemos descargar, **Google Cloud Messaging**.

GCMBaseIntentService es una clase que, a su vez, hereda de la clase *Service*. Clase que tiene la peculiaridad de mantenerse en ejecución incluso con la aplicación cerrada.

En nuestra clase, la cual llamaremos *GCMIntentService* deberemos implementar 4 métodos heredados de la clase *GCMBaseIntentService:*

- **onError:** Este método será ejecutado cuando haya algún error en el registro GCM.
- **onMessage:** El método más importante de la clase y el que más utilizaremos. Este método será ejecutado cada vez que el servicio GCM nos envíe una notificación, que, a su vez, habremos indicado nosotros desde el servidor. De esta manera, en cualquier momento, este o no la aplicación en ejecución, podremos ejecutar este método.
  Este método será utilizado en mayor parte para, cada vez que recibamos un mensaje, mandar una notificación al dispositivo para avisar al usuario de que tiene un nuevo mensaje; e introducir este mensaje en la base de datos de la aplicación, de tal forma que cuando accedamos a la conversación solo tengamos que consultar los mensajes de nuestra propia base de datos y no la del servidor.
- **onRegistered:** Este método será ejecutado cuando nos registremos con éxito en el sistema GCM. Nos registraremos cuando creemos la cuenta de usuario y, cuando lo hagamos, este método será ejecutado. En él, recibiremos como parámetro la GCM key (el identificador de usuario del servicio), la cual deberemos mandar al servidor, para que este conozca que identificador mandar a GCM para que envíe los mensajes.
- **onUnregistered:** Este método será ejecutado cuando se borra el dispositivo del servicio GCM.

5.5.3 Inicio de conversaciones

Para acceder a la pantalla de conversación podremos hacerlo de diferentes maneras, dependiendo si esa conversación está iniciada o no. En el caso de que tengamos una conversación iniciada, podremos acceder directamente desde la pantalla de selección de conversación.

En caso contrario de que no dispongamos de una conversación activa con una persona o dicha conversación este muy abajo en la lista y no podamos encontrarla, deberemos acceder a la pantalla de selección de contactos mediante un botón en la parte superior de la pantalla actual. En esta pantalla aparecerán todos nuestros contactos en la aplicación y dispondremos en la parte superior de un buscador para encontrar más rápidamente el contacto que buscamos.

En el caso de acceder a la conversación de un grupo deberemos acceder únicamente desde la selección de conversaciones o crear uno nuevo como se explicó con anterioridad en este documento.

5.5.4 Estado del mensaje

En conversaciones de únicamente dos personas, para conocer el estado de un mensaje enviado, es decir, si el mensaje ha sido visto por el receptor; lo haremos de la siguiente manera. Todo mensaje una vez es generado será marcado como enviado pero no visto.

Cuando nosotros tenemos mensajes de otro usuario sin ver y accedemos a su conversación, estos mensajes serán marcados como vistos, y el servidor será notificado para que modifique esta información en el servidor.

Cuando estemos dentro de una conversación, en el caso de que haya mensajes enviados por nosotros marcados como "no vistos", se solicitarán peticiones al servidor para cada mensaje marcado como tal, para verificar si el estado del mensaje ha cambiado o no. En el caso de que haya cambiado, el servidor responderá a la aplicación para que esta modifique su base de datos.

5.6 Gestión de mensajes

Para gestionar la información y los mensajes que recibamos en nuestra aplicación, podremos marcarlos como mensajes importantes para nosotros exclusivamente o para el grupo entero, en el caso de ser una conversación grupal y tengamos permiso para realizar esta acción.

Para hacerlo, mantendremos pulsado el mensaje en la pantalla y elegiremos la marcación correspondiente.

Dentro de cada conversación, podremos filtrar los mensajes por importancia: todos, importancia personal, importancia grupal. De esta forma solo mostraremos los mensajes que deseemos y obtendremos una visión más clara de la importancia de la conversación. En cada una de estas vistas también podremos realizar búsquedas entre los mensajes para encontrar información mucho más rápido.

5.7 Aviso de conversaciones temáticas

Por medio de un servicio, cada usuario registrado en un grupo con fecha de inicio, será avisado cuando dicha fecha se aproxime.

De esta forma todos los usuarios podrán crear conversaciones para una determinada fecha y para un determinado tema, invitando a los usuarios a estos grupos, para que puedan participar en las mismas.

Estos avisos podrán ser personalizables, tanto la forma en la que se avise como el tiempo previo al inicio de la conversación.

Capítulo 6
Análisis de costes

En este apartado analizaremos los costes estimados que supone la elaboración del proyecto. Estos costes engloban costes de personal, costes materiales, costes de subcontratación de tareas y resto de costes que puedan surgir durante el desarrollo del proyecto. Estos serán recogidos y acumulados finalmente en una tabla donde se calculará el presupuesto total del proyecto.

6.1 Costes de personal

Nombre y apellidos	Categoría	Dedicación (hombres mes) [a)]	Coste hombre mes	Coste (Euro)
	Analista del proyecto	1	2.159,00	2.159,00
Sergio Chicharro Sobrino	Programador	4	2.159,00	8.636,00
	Gestor de proyecto	2	2.159,00	4.318,00
			Total	15.113,00

[a)] 1 Hombre mes = 131,25 horas. Máximo anual de dedicación de 12 hombres mes (1575 horas)
Máximo anual para PDI de la Universidad Carlos III de Madrid de 8,8 hombres mes (1.155 horas)

6.2 Costes materiales

Descripción	Coste (Euro)	% Uso dedicado proyecto	Dedicación (meses)	Período deprecación	Coste imputable [d)]
Ordenador para desarrollo	1.230,00	100	4	60	82,00
LG Google Nexus 4	350,00	100	4	16	87,50
Moto G	175,00	100	4	16	43,75
				Total	213,25

[d)] Fórmula de cálculo de la Amortización:

$$\frac{A}{B} xCxD$$

A = nº de meses desde la fecha de facturación en que el equipo es utilizado
B = periodo de depreciación (60 meses)
C = coste del equipo (sin IVA)
D = % del uso que se dedica al proyecto (habitualmente 100%)

6.3 Subcontratación de tareas

Descripción	Empresa	Coste imputable
	Total	0,00

6.4 Otros gastos

Esta tabla incluye los gastos no tenidos en cuenta en las tablas anteriores.

Descripción	Empresa	Coste imputable
	Total	0,00

6.5 Costes totales

Presupuesto Costes Totales	Presupuesto Costes Totales (€)
Personal	15.113,00
Amortización	214,00
Subcontratación de tareas	0
Otros gastos	0
Costes Indirectos	3.065,40
Total	**18.392,40**

Capítulo 7
Conclusiones y trabajos futuros

7.1 Conclusiones

Como producto de este proyecto hemos obtenido una aplicación de mensajería instantánea accesible para dispositivos Android que cumple con las necesidades de

accesibilidades requeridas y la orientación académica de la aplicación para su uso en este entorno.

Se ha desarrollado un sistema de almacenamiento de datos bien estructurado lo que permitirá la fácil expansión o modificación de estos si así es requerido en un futuro.

El código de la aplicación web y la aplicación Android ha sido igualmente estructurado y dividido en clases lo que provoca una clara y limpia presentación del código, facilitando su modificación si esto fuese necesario.

La aplicación ofrece una alternativa accesible a todos los sistemas de mensajería móviles actuales, permitiendo una amplia personalización en muchos aspectos de su interfaz, haciendo su uso más cómodo y fácil tanto para usuarios corrientes como para usuarios con algún tipo de dificultad para utilizar aplicaciones comunes.

La aplicación Android presenta una interfaz limpia, intuitiva y sin sobrecargas de información que ayuda al usuario a utilizarla con rapidez y facilidad. Además, se ha intentado optimizar la velocidad ejecución y el tiempo de respuesta de la aplicación, se han reducido las peticiones al servidor al mínimo para prevenir un excesivo flujo de datos innecesario y se han utilizado patrones predecibles y comunes en otro tipo de aplicaciones para reducir la incertidumbre del efecto de estos patrones en la aplicación y ajustar el programa al comportamiento habitual del usuario.

Por último, podemos resaltar que la aplicación es de fácil instalación, pues será suficiente con ejecutar el instalador de la misma en nuestro dispositivo Android para que esta se instale en nuestro equipo automáticamente y esté lista para su uso.

7.2 Trabajos futuros

Personalmente, este proyecto ha sido de gran utilidad, pues Android era una tecnología que desconocía hasta el momento a nivel de desarrollador y con las habilidades y conocimientos obtenidos durante la realización del proyecto he podido ampliar mi formación y mis posibilidades en vistas de un futuro laboral próximo.

Podré realizar mis proyectos personales en Android si así lo deseo, pues es una tecnología muy potente una vez tienes la destreza necesaria para programar con ella. Android es de código abierto, y esto es de gran utilidad al incluir otras librerías y APIs a tu propia aplicación, para hacerla mucho más potente, y una vez lista, si así lo deseas puedes publicarla para el resto de la comunidad que utilice Android de una forma muy fácil y rápida, e incluso, sacar algún beneficio de la aplicación.

En cuanto a la aplicación desarrollada podríamos futuramente ampliar y expandir su funcionalidad haciéndola más útil para los usuarios y convirtiéndola en una opción alternativa de más importancia con respecto al resto de aplicaciones de mensajería instantánea.

Podremos adaptar esta aplicación a otras plataformas como iOS, Windows Phone, aplicación web de navegador o aplicación de escritorio para distintos sistemas operativos. De esta forma la aplicación podrá ser utilizada por usuarios que no dispongan de un dispositivo Android, generalizando el uso de la aplicación.

Por último, podremos adaptar y añadir a la aplicación nueva funcionalidad que el sistema operativo de Android nos permita implementar con sus nuevas actualizaciones.

Capítulo 8
Glosario de términos

Con el fin de aclarar los tecnicismos y abreviaturas utilizados durante el desarrollo del presente documento, a continuación se presenta una lista con la explicación de cada uno de estos términos:

- **.NET.** Framework desarrollado por Microsoft que permite el desarrollo de software en varios lenguajes como C, VisualBasic, C#...
- **ADT.** Plugin para la IDE de Eclipse que establece un entorno de trabajo adaptado al desarrollo de aplicaciones Android.
- **Android.** Sistema operativo basado en Linux orientado y diseñado principalmente para dispositivos móviles.
- **Apache.** Servidor web HTTP de código abierto.
- **API.** Conjunto de funciones y procedimientos de una determinada biblioteca, los cuales podemos incluir y utilizar en otros proyectos.
- **C.** Lenguaje de programación orientado al desarrollo de sistemas operativos. No es un lenguaje orientado a objetos.
- **C#.** Lenguaje de programación basado en C orientado a objetos. Desarrollado por Microsoft como parte de la plataforma .NET.
- **C++.** Lenguaje de programación derivado de C que soporta la orientación a objetos.

- **Clave primaria.** En bases de datos, campo de una tabla que sirve para identificar una determinada fila.
- **Clave foránea / Clave ajena.** En bases de datos, la clave ajena es un campo utilizado para establecer relaciones entre tablas. Esta clave foránea coincidirá con la clave primaria de otra tabla, lo que relacionará ambas filas de las tablas.
- **Eclipse.** Programa utilizado para el desarrollo de IDE's. Además incluye el IDE JDK que nos permitirá desarrollar aplicaciones en Java. Otros IDE's pueden ser incluidos e instalados en este software para habilitar la programación en otros lenguajes.
- **Framework.** Entorno o estructura informática que incluye herramientas que nos permiten desarrollar software en determinados lenguajes.
- **GCM.** Google Cloud Messaging es un servicio de Android desarrollado y distribuido por Google que nos permite enviar notificaciones a un dispositivo sin necesidad de que el dispositivo se encuentre ejecutando nuestra aplicación.
- **IDE.** Un IDE o entorno de desarrollo integrado está compuesto por un conjunto de herramientas para facilitar el desarrollo de software.
- **iOS.** Sistema operativo diseñado para dispositivos móviles desarrollado por Apple Inc.
- **JDBC.** Java Database Connectivity es una API que permite la conexión a bases de datos desde el propio código Java.
- **JDK.** Software que proporciona herramientas necesarias para el desarrollo de aplicaciones en Java.
- **JSP.** Java Server Pages es una tecnología para desarrolladores que permite el desarrollo de páginas web basadas en HTML, XML...
- **Login.** Término que se refiere a la acción de comprobar la autenticidad de un usuario frente a un sistema informático y se da acceso al mismo.
- **Netbeans.** IDE orientado principalmente para el desarrollo en Java.
- **Perl.** Lenguaje de programación basado en C.
- **Plugin.** Complemento a una aplicación que extiende su funcionalidad o añade herramientas para su uso.
- **Preview.** Fase temprana del desarrollo de un sistema informático. Suele presentar fallos y bugs y su funcionalidad es reducida.
- **SDK.** SDK o Software Development Kit es un conjunto de herramientas utilizadas para el desarrollo de software para un lenguaje de programación determinado.
- **Visual Basic.** Lenguaje de programación dirigido por eventos desarrollado por Microsoft.
- **XAMPP.** Herramienta de software libre que permite el despliegue de una base de datos MySQL, un servidor web Apache e intérpretes de lenguajes PHP y Perl.

- **XML.** Lenguaje de marcas desarrollado por el W3C (World Wide Web Consortium) utilizado para almacenar datos de forma estructurada.

Capítulo 9
Bibliografía

1. Frank Ableson, Robi Sen y Chris King. Android. Guía para desarrolladores. ANAYA MULTIMEDIA /MANNIN. ISBN: 978-84-415-2958-8

2. Wei-Meng Lee. Android 4. Desarrollo de aplicaciones. ANAYA MULTIMEDIA / WROX. ISBN: 978-84-415-3197-0

3. Android developers [en línea] http://developer.android.com/intl/es/index.html

4. Introducción a la Accesibilidad Web [en línea] http://www.w3c.es/Traducciones/es/WAI/intro/accessibility

5. Android Accesible [en línea] http://android-accesible-espanol.blogspot.com.es/

6. PHP: Manual de PHP [en línea] http://www.php.net/manual/es/

7. MySQL :: MySQL 5.0 Reference Manual [en línea] https://dev.mysql.com/doc/refman/5.0/es/

8. Google Cloud Messaging. Parte I: Introducción | Belén Cruz [en línea] http://belencruz.com/2013/01/google-cloud-messaging-parte-i-introduccion/

9. Android | Examples Java Code Geeks [en línea] http://examples.javacodegeeks.com/android/

10. W3Schooles Online Web Tutorials [en línea] http://www.w3schools.com/

11. Notificaciones Push Android: Google Cloud Messaging (GCM). Implementación Cliente (Nueva Versión) RSS de los comentarios [en línea] http://www.sgoliver.net/blog/?p=4126

12. Aprendiendo Android VII: Los hilos - Threads [en línea] http://www.elandroidelibre.com/2010/09/aprendiendo-android-vii-los-hilos-threads.html

13. Tutoriales - HTML 5 -: Android: Thread (Hilo) y Handler, proceso en segundo plano [en línea] http://www.tutorialeshtml5.com/2012/05/android-thread-hilo-y-handler-proceso.html

14. Las notificaciones de la barra de estado - Diploma de Especialización en desarrollo de aplicaciones para Android [en linea] http://www.androidcurso.com/index.php/curso-android-fundamentos/tutoriales-android-fundamentos/38-unidad-8-servicios-notificaciones-y-receptores-de-anuncios/161-las-notificaciones-de-la-barra-de-estado

15. Android Speech to Text API. Speech to Text using RecognizerIntent [en línea] http://viralpatel.net/blogs/android-speech-to-text-api/

16. Wikipedia, la enciclopedia libre [en línea] http://es.wikipedia.org

17. La accesibilidad para invidentes en Android 4.0 [en línea] http://www.xatakandroid.com/sistema-operativo/la-accesibilidad-para-invidentes-en-android-40

18. Programación Android: Interfaz gráfica - Componentes gráficos y Eventos - El Baúl del Programador [en línea] http://elbauldelprogramador.com/programacion-android-interfaz-grafica_25/

19. Diseño visual | Diseñando apps para móviles [en línea] http://www.appdesignbook.com/es/contenidos/diseno-visual/

20. Desarrollo de interfaces dinámicas de usuario con Android y XML [en línea] http://www.ibm.com/developerworks/ssa/xml/tutorials/x-andddyntut/

21. Fragments dinámicos en Android | SekthDroid [en línea] http://sekthdroid.wordpress.com/2013/02/01/fragments-dinamicos-en-android/

22. Curso Android: Trabajando con imágenes (cámara y galería) [en línea] http://www.maestrosdelweb.com/editorial/curso-android-trabajando-con-imagenes/

23. Manejo de datos BLOB con PHP y MySQL. Programación en Castellano. [en línea] http://www.programacion.com/articulo/manejo_de_datos_blob_con_php_y_mysql_194

24. Working with Images in Your Android App Development - Developer.com [en línea] http://www.developer.com/ws/android/programming/Working-with-Images-in-Googles-Android-3748281.htm

25. Memorias de un aprendiz de PHP [en línea] http://www.rinconastur.com/php/php71.php

26. SQLite Documentation [en línea] http://www.sqlite.org/docs.html

27. Send and Receive JSON between Android and PHP Web Service - CodeProject [en línea] http://www.codeproject.com/Articles/267023/Send-and-receive-json-between-android-and-php

28. JSON III – Gestionar JSON en PHP | Geeky Theory [en línea] http://geekytheory.com/json-iii-gestionar-json-en-php/

29. Leyendo servicios web desde Android: JSON | Nosinmiubuntu | Ubuntu en concreto, GNU/Linux en general [en línea] http://www.nosinmiubuntu.com/2012/02/leyendo-servicios-web-desde-android.html

30. Parseando JSON en Android | Androcode [en línea] http://androcode.es/2012/05/parseando-json-en-android/

31. How to connect to MySQL database using PHP [en línea] http://webcheatsheet.com/php/connect_mysql_database.php

32. Guardar leer preferencias de configuración aplicación Android SharedPreferences Proyecto AjpdSoft [en línea] http://www.ajpdsoft.com/modules.php?name=News&file=article&sid=556

33. Almacenamiento en una base de datos SQLite [en línea] http://www.javaya.com.ar/androidya/detalleconcepto.php?codigo=145&inicio=

34. Blog - Notificaciones Push en Android (Introducción) - IOS - Android - Desarrollo Web - Granada [en línea] http://www.appandweb.es/blog/android/notificaciones-push-en-android-(introduccion)/

35. Google Cloud Messaging (CGM) y nodejs | cartuchoGL [en línea] http://cartucho.gl/google-cloud-messaging-cgm-y-nodejs/

36. Accessibility basics - Android Accessibility [en línea] http://eyes-free.googlecode.com/svn/trunk/documentation/android_access/basics.html

Capítulo 10
Anexos

Complementando el presente documento se adjuntan:

- Imagen del diagrama de Gantt que representa la planificación temporal del proyecto.

- Guía de desarrollo de la que partimos como ejemplo.

- Resumen de este documento en inglés.

- Manual de usuario de la aplicación.

- Manual de desarrollador de la aplicación.